

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : WAN RUSLAN BIN W YUSOFF
Date of Birth : 11 SEPTEMBER 1956
Title : REALTIME INTERPOLATION OF PARAMETRIC
: CURVES WITH CHORD-ERROR AND
: FEEDRATE CONSTRAINTS.
Academic session : SEMESTER II 2022/2023

I declare that this thesis is classified as:

- CONFIDENTIAL** (Contains confidential information under the Official Secret Act 1997)*
 RESTRICTED (Contains restricted information as specified by the organization where research was done)*
 OPEN ACCESS I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following right:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

(Student's Signature)

(Supervisor's Signature)

New IC: 560911035067

Assoc Prof Fadhlur Rahman

Date: September 16, 2024

bin Mohd Romlay

Date: September 16, 2024

NOTE: *If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

THESIS DECLARATION LETTER (OPTIONAL)

Librarian,
Perpustakaan Universiti Malaysia Pahang,
Universiti Malaysia Pahang,
Lebuhraya Tun Razak,
26300, Gambang, Kuantan.

Dear Sir,

CLASSIFICATION OF THESIS AS RESTRICTED

Please be informed that the following thesis is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Author's Name : WAN RUSLAN BIN W YUSOFF

Thesis Title : Realtime interpolation of parametric curves with chord-error
and feedrate constraints.

Reasons:

- 1.
- 2.
- 3.

Thank you.

Yours faithfully,

(Supervisor's Signature)

Date:

Stamp:

Note: This letter should be written by the supervisor, addressed to the Librarian, *Perpustakaan Universiti Malaysia Pahang* with its copy attached to the thesis.



SUPERVISOR'S DECLARATION

We hereby declare that we have checked this thesis and in our opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Doctor of Philosophy of Engineering in Mechatronics and System Design.

(Supervisor's Signature)

Name of Supervisor: Fadhlur Rahman bin Mohd Romlay

Position: Associate Professor

Date: September 16, 2024

(Co-supervisor's Signature)

Name of Supervisor: Yashwant Prasad Singh

Position: Professor

Date: September 16, 2024



STUDENTS'S DECLARATION

I hereby declare that the work in this thesis is my own for quotations and summaries which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

(Student's Signature)

Name: Wan Ruslan bin W Yusoff

ID number: PFD18001

Date: September 16, 2024

**REALTIME INTERPOLATION OF PARAMETRIC CURVES WITH CHORD-ERROR
AND FEEDRATE CONSTRAINTS**

WAN RUSLAN BIN W YUSOFF

Thesis submitted in fulfilment of the requirements
for the award of the degree of
Doctor of Philosophy of Engineering in Mechatronics and System Design

Faculty of Automotive and Mechanical Engineering
UNIVERSITI MALAYSIA PAHANG

XXXX====; OCTOBER 2024 (TO CHANGE)

ACKNOWLEDGEMENTS

In the name of Allah, Most Gracious and Most Merciful. Indeed all praises be unto Allah, and we praise Him, and we seek help from Him, and we seek forgiveness from Him. We seek refuge from Him from the evil of ourselves, and from the evil of our actions. Whomsoever Allah guides, none can misguide. And whomsoever He leaves astray, none can guide. May the Peace, Mercy and Blessings of Allah be upon you.

Foremost, I thank Allah, the Most Glorious and Most High, for granting me the opportunity to tread down the unknown trail on this research journey. I wish to also convey my sincerest gratitude to all people who have directly or indirectly, or will be involved with me on this journey. There are just too many people to mention.

Specially to Prof. Yashwant Prasad Singh, perceived as many personalities to me: As my guru, teacher, mentor, advisor, supervisor and a dear friend, I am very grateful to him for his unimaginable faith, persistence and enthusiasm in encouraging, guiding, and sharing with me knowledge throughout the many wonderful years of our academic acquaintance. As my Supervisor, his advice was simple, "Your PhD study should be exciting and fun." And that is true. We spent long hours and fruitful discussions on almost unlimited topics, from philosophy, politics, religion, and family to the hard sciences, computer science, and engineering. We also made a pact to remain as lifelong friends, Insya Allah, God Willing. With internet facilities today, we are constantly in communication.

As a tribute to Assoc. Prof. Dr. Fadhlur Rahman, my direct supervisor, I am eternally indebted to him for his sincere trust, unbelievable patience, constant guidance and timely assistance with all matters, including research equipment, resources and many other administrative needs of the university. To my brother, Prof. Ir. Dr. Wan Azhar, I undyingly appreciate his challenge that I should eventually get a doctorate. To my son, Abdulazeez, I thank him adoringly for his unequivocal faith, continuous encouragement and financial assistance in my endeavor.

To my family, especially my wife, my sons and daughter, I thank them affectionately for their unshakable love, utmost patience, undivided support and unwavering understanding during the long hours and sleepless nights I went through. The coffee and snacks were never interrupted. For smooth English writing, my wife is also my bouncer and editor.

To those in my extended family with PhDs, I thank them admiringly for their strange looks and jokes. One senior poked fun with a comment at me. He said, "Considered to be the most intelligent among us, he does not have a doctorate. Hahaha." My cynical response was, "Is it not inspiring that for many years I have been doing work of people

with PhDs, but without a doctorate myself? I feel, it is certainly humbling but yet assuring being accorded that level of belief in my ability.”

To my friends, I kindly thank them for all support and encouragement rendered to me during my research journey. To Multimedia University (MMU), I thank them for the opportunity provided to me for teaching, research and those unforgettable interactions with Prof Singh and staff at the university. To University Malaysia Pahang (UMP), I thank them for accepting me as a research student (at the age of mid 60s) and for the generosity of providing research equipment and resources.

Finally, I again praise Allah, for the invaluable gifts of health, time and clear state of mind, without which, I would not have been able to go through this arduous and exhausting journey. There is always a purpose in everything. Thinking of it, I recalled the motto of a local university, ”To Allah and Mankind”. Without hesitation, I say, this is exactly the one for me. God Willing, may Allah grant me this deserving wish. In closing, I wish to share the following passages from Allah, the All-Knowing and All-Mighty.

And in His Providence are the keys of the Unseen; none knows them except He. And He knows whatever is in the land and the sea. And in no way does a leaf fall down, except that He knows it, and not a grain in the darkness of the earth, not a thing wet or dry, except that it is in an evident Book. Whoever submits his whole self to God and is a doer of good, he has grasped indeed the most trustworthy handhold. And with God rests the end and decision of all affairs. Verily, When He (Allah) intends a thing, his command is “Be ! and it is !”.

(*Combined verses Al-Quran, Chapters Al-An'am 6:59, Lukman 31:22 and Ya-Sin 36:82*)

A recruiter once told this famous story. He was invited to attend a college team competition where autonomous underwater vehicles must avoid obstacles before reaching the finish line. The competition was held in a swimming pool. After the competition, seeing that the runner up team was sad, he approached them and asked a question, ”How much did you spend on your project?” The answer he got was 20K. He remarked, ”Oh yeah. But the champion spent 100K. Now who is better?”

Alhamdulillah. With all my love. Always.

Wan Ruslan Yusoff
UMP Pekan, Pahang
September 16, 2024

ABSTRAK

Mengakui bahawa trend pemesinan CNC adalah ke arah lengkung NURBS, makalah ini menyajikan interpolasi masa nyata bagi kelas lengkung NURBS yang sememangnya parametrik. Sepuluh lengkung NURBS 2-dimensi yang berbeza dipilih berdasarkan pelbagai ciri, bentuk dan dimensi. Ciri lengkung merangkumi variasi, ukuran dimensi x dan y, asal atau tidak berpusat, lengkung tertutup atau terbuka, dengan bilangan gelung yang berbeza, dengan segmen cembung atau cekung, dengan putaran tajam atau licin, dan simetri pantulan yang berbeza mengenai paksi x dan y.

Algoritma interpolasi masa nyata apabila diterapkan pada semua lengkung yang dipilih secara eksklusif dan serentak memenuhi kedua-duakekangan yang dirancang, yang meliputi, kadar suapan dan toleransi kord-errornya. Profil feedrate yang dihasilkan di seluruh jalan lengkung berterusan dan lancar. Kekangan feedrate merangkumi persamaan dinamik untuk parameter mesin CNC yang dibenarkan seperti halaju paksi maksimum dan minimum, dan pecutan paksi maksimum dan minimum. Kekangan kord-error terdiri daripada sifat geometri dan kinematik dari sepuluh (10) lengkung parametrik yang berbeza, seperti selekoh dan putaran tajam. Algoritma yang dihasilkan dapat dijalankan baik dalam mod dalam talian masa nyata dengan menggerakkan mesin CNC secara langsung, atau dalam mod luar talian dengan menggunakan fail G-kod RS274/NGC.

ABSTRACT

Acknowledging that the trend in CNC machining is toward NURBS curves, this paper presents the realtime interpolation of a class of NURBS curve which is inherently parametric. Ten different 2-dimensional NURBS curves were selected based on varying features, shapes and dimensions. The curve characteristics cover variations, in size of x and y dimensions, of origin or not-origin centered, of closed or open ended curves, with different number of loops, with convex or concave segments, with sharp or smooth turns, and different reflection symmetry about the x and y axes.

The realtime interpolation algorithm when applied to all of the selected curves exclusively and simultaneously satisfy both of its designed constraints, that covers, its feedrate and its chord-error tolerance. The resulting feedrate profiles throughout the entire path of the curves are continuous and smooth. The feedrate constraints comprise dynamic equations for allowable CNC machine parameters like the maximum and minimum axial velocities, and the maximum and minimum axial accelerations. The chord-error constraints comprise geometric and kinematic properties of the ten(10) different parametric curves, like bends and sharp turns. The resulting algorithm can be executed both in a realtime, online mode by driving the CNC machine directly, or in an offline mode by using a stored RS274/NGC G-code file.

TABLE OF CONTENTS

DECLARATION

TITLE PAGE

ACKNOWLEDGEMENTS	i
-------------------------	----------

ABSTRAK	iii
----------------	------------

ABSTRACT	iv
-----------------	-----------

TABLE OF CONTENTS	v
--------------------------	----------

LIST OF TABLES	xxv
-----------------------	------------

LIST OF FIGURES	xxviii
------------------------	---------------

LIST OF ABBREVIATIONS	xlvii
------------------------------	--------------

CHAPTER 1 INTRODUCTION	1
-------------------------------	----------

1.1 Introduction	1
1.2 Problem statement	2
1.3 Motivation	3
1.4 Brief description of this thesis	4
1.5 About end results of this thesis	6
1.6 Scope of work for this thesis	7
1.7 Organization of this thesis document	9

CHAPTER 2 LITERATURE REVIEW	11
------------------------------------	-----------

2.1 Parametric Representation of curves and surfaces	11
2.2 Continuity of curves and surfaces	12
2.3 Computerized Numerical Control (CNC) Systems	13
2.4 Advantages of Parametric Representations in CNC	18
2.5 Interpolation of parametric curves	19
2.6 Previous works on parametric interpolation	20
2.7 Related CNC work by the author	22
2.8 Comparisons previous works with this thesis	26

2.9	NURBS and G-Code programming	27
2.10	Programming Languages for NURBS	28
CHAPTER 3 METHODOLOGY		30
3.1	Parametric Curve Interpolation	30
3.2	Selection of parametric curves	30
3.3	Computing challenges in the algorithm	31
3.4	Characteristics of selected curves	32
3.5	Curve shapes and curve equations	33
3.6	Links to parametric curves	35
3.7	Direction of travel in parametric curves	49
3.8	Chord-error concept	51
3.9	Chord-error minimization	51
3.10	Feedrate maximization	52
3.11	Chord-error and feedrate constraints	52
3.12	Brief on algorithm strategy and design	52
3.13	Radius of Curvature $\rho(u)$	55
3.14	Chord-error $\epsilon(u)$	55
3.15	Feedrate or Velocity $V(u)$	57
3.16	First order Taylor's approximation $u_{\text{next}}(u)$	58
3.17	Second order Taylor's approximation $u_{\text{next}}(u)$	59
3.18	Next interpolation point calculation	61
3.19	Feedrate limit calculations	62
3.19.1	Feedrate Limit 1	63
3.19.2	Feedrate Limit 2	63
3.19.3	Feedrate Limit 3	64
3.19.4	Feedrate Limit 4	65
3.20	Feedrate rising S-curve	66
3.21	Flowchart of main interpolation program	67
3.22	Feedrate Limit algorithm regions	70

3.23	Feedrate falling S-curve	80
3.24	Acceleration constraint lambda safety factor	81
3.25	Four(4) Algorithm Performance Metrics	82
	3.25.1 Arc-Length Approximate calculation	84
	3.25.2 Chord-Length Exact calculation	85
	3.25.3 Arc-Theta Approximate calculation	86
	3.25.4 Arc-Area Approximate calculation	87
3.26	Software engineering practice	88
3.27	Design of Algorithm Codes	89
	3.27.1 Algorithm Text Reports	89
	3.27.2 Algorithm Functional Organization	91
	3.27.3 Algorithm and runtime parameters	95
3.28	Working environment setup	97
3.29	System Hardware Environment	98
3.30	Software Environment	99
	3.30.1 Operating System	99
	3.30.2 Programming Software Languages	99
	3.30.3 Reporting Software Applications	100
	3.30.4 Specialized Software Applications	100
3.31	CNC System Setup	101
CHAPTER 4 RESULTS AND DISCUSSIONS		110
4.1	CHAPTER ORGANIZATION	110
	4.1.1 Result comparisons with Published Reference Paper	110
	4.1.2 Algorithm Executions	110
	4.1.3 Teardrop Curve for Illustration	111
	4.1.4 Results of Teardrop curve	111
	4.1.5 Notable Results for Rest of Curves	111
	4.1.6 Overall Execution Results	111
	4.1.7 Summary of Results Chapter	112

4.2	COMPARISONS AGAINST PUBLISHED PAPER	113
4.2.1	Result comparisons with Published Reference Paper	113
4.2.2	Display comparison results	114
4.2.3	Discussion on comparative results	114
4.2.4	Validation of Teardrop curve on CNC machine	120
4.3	ALGORITHM EXECUTIONS	122
4.3.1	Total algorithm executions	122
4.3.2	Algorithm revisions	122
4.3.3	Illustrative example Teardrop curve execution	123
4.3.4	Determination of acceptable lamda	124
4.3.5	Examples of acceleration jitters (landscape)	126
4.3.6	Results for finding acceptable lamda	131
4.3.7	Rechecking acceptable lamda	132
4.4	TEARDROP CURVE FOR ILLUSTRATION	147
4.4.1	Teardrop parametric curve	147
4.4.2	Rest of other nine(9) parametric curves	147
4.5	RESULTS OF TEARDROP CURVE	151
4.5.1	Plot of Teardrop curve	151
4.5.2	Teardrop Direction of Travel	151
4.5.3	Teardrop Perspective View 3D in LinuxCNC-Axis	153
4.5.4	Teardrop Radius of Curvature	154
4.5.5	Teardrop 1st and 2nd Order Taylor's Approximation	156
4.5.6	Teardrop (1st minus 2nd) Order Taylor's Approximation	156
4.5.7	Teardrop Chord-Error Absolute Constraint	158
4.5.8	Teardrop Four(4) Feedrate Limit Components Profile	158
4.5.9	Teardrop FRate Command, FRate Limit, Current FRate	161
4.5.10	Teardrop Current Feedrate Absolute Constraint	161
4.5.11	Teardrop Histogram distribution of interpolated points	163
4.5.12	Teardrop Table distribution of interpolated points	163

4.5.13	Teardrop Rising Current Feedrate Profile	165
4.5.14	Teardrop Falling Current Feedrate Profile	165
4.5.15	Teardrop FC10, FC20, TC30 & FC40 Running Feedrates	167
4.5.16	Teardrop Color-coded Running Feedrates	170
4.5.17	Teardrop-Table FC 10, 20, 30 & 40 Performance data	173
4.5.18	Teardrop Algorithm Performance and its metrics	178
4.5.19	Teardrop Tangential Acceleration Run Profiles	183
4.6	PERFORMANCE SUMMARY REST OF CURVES	189
4.6.1	Circle-Table Summary Performance data link [4.10]	189
4.6.2	Ellipse-Table Summary Performance data link [4.11]	189
4.6.3	Butterfly-Table Summary Performance data link [4.12]	189
4.6.4	Snailshell-Table Summary Performance data link [4.13]	189
4.6.5	Skewed-Astroid-Table Summary Performance data link [4.14]	189
4.6.6	Ribbon-10L-Table Summary Performance data link [4.15]	189
4.6.7	Ribbon-100L-Table Summary Performance data link [4.16]	189
4.6.8	AstEpi-Table Summary Performance data link [4.17]	189
4.6.9	SnaHyp-Table Summary Performance data link [4.18]	189
4.7	NOTABLE RESULTS REST OF CURVES	199
4.7.1	Algorithm validation and verification of Circle curve	199
4.7.2	Algorithm validation and verification of Ellipse curve	200
4.7.3	Case Sum-Arc-Length (SAL) less than Sum-Chord-Length (SCL)	202
4.7.4	SnaHyp curve algorithm machine epsilon problems	210
4.8	OVERALL EXECUTION RESULTS	216
4.8.1	Feedrate Command (FC) and Lamda safety factor (LSF)	216
4.8.2	Algorithm Validation and Verification (V & V)	217
4.8.3	LinuxCNC-Axis Simulation Run Validation (SRV)	218
4.8.4	LinuxCNC-Axis Real Run Validation (RRV)	219
4.8.5	Overall Total Interpolated Points (TIP)	221
4.8.6	Overall Total Sum-Arc-Length (SAL)	224

4.8.7	Overall Total Sum-Chord-Length (SCL)	227
4.8.8	Overall Percentage Difference between SAL and SCL	230
4.8.9	Overall Total Sum-Chord-Error (SCE)	234
4.8.10	Overall Chord-Error-Efficiency (CEE)	237
4.8.11	Overall Total Sum-Arc-Area (SAA)	240
4.8.12	Overall Arc-Area-Efficiency (AAE)	244
4.9	SUMMARY OF RESULTS CHAPTER	247
CHAPTER 5 CONCLUSION		250
5.1	Conclusions of the research	250
5.2	Performance accuracy of the algorithm	251
5.3	Knowledge contributions	252
5.4	Recommendations for future work	252
REFERENCES		253
APPENDICES		257
.1	APPENDIX CHAPTER 3	258
.1.1	About Side Effects in Computations	258
.1.2	Realtime Computing Environment	260
.1.3	Cyclictest for computer latency measurements	262
.1.4	CPU Latency Performance Summary	265
.2	APPENDIX CHAPTER 4	266
.2.1	Ellipse Perimeter using Ramanujan Approximation	266
.2.2	Calculation of machine epsilon C code	268
.2.3	Ranges of floating-point numbers in C code	269
.2.4	Machine epsilon affecting addition and subtraction operations	270
.2.5	Resolving machine epsilon issue	271
.2.6	LinuxCNC Validation of Curves	272
.3	APPENDIX CIRCLE CURVE	284
.3.1	Plot of Circle curve [15]	284

.3.2	Circle Radius of Curvature [16]	284
.3.3	Circle Validation in LinuxCNC [17]	284
.3.4	Circle Direction of Travel 3D [18]	284
.3.5	Circle First and Second Order Taylor's Approx [19]	284
.3.6	Circle First minus Second Order Taylor's Approx [20]	284
.3.7	Circle Separate First and Second Order Taylor's Approx [21]	284
.3.8	Circle Separation SAL and SCL [22]	284
.3.9	Circle Chord-error in close view 2 scales [23]	284
.3.10	Circle Four Components Feedrate Limit [24]	284
.3.11	Circle FrateCommand FrateLimit and Curr-Frate [25]	284
.3.12	Circle FeedRateLimit minus CurrFeedRate [26]	284
.3.13	Circle FC20-Nominal X and Y Feedrate Profiles [27]	284
.3.14	Circle FC20 Nominal Tangential Acceleration [28]	284
.3.15	Circle FC20 Nominal Rising S-Curve Profile [29]	284
.3.16	Circle FC20 Nominal Falling S-Curve Profile [30]	284
.3.17	Circle FC10 Colored Feedrate Profile data ngcode [31]	284
.3.18	Circle FC20 Colored Feedrate Profile data ngcode [32]	284
.3.19	Circle FC30 Colored Feedrate Profile data ngcode [33]	285
.3.20	Circle FC40 Colored Feedrate Profile data ngcode [34]	285
.3.21	Circle FC10 Tangential Acceleration [35]	285
.3.22	Circle FC20 Tangential Acceleration [36]	285
.3.23	Circle FC30 Tangential Acceleration [37]	285
.3.24	Circle FC40 Tangential Acceleration [38]	285
.3.25	Circle FC20 Nominal Separation NAL and NCL [39]	285
.3.26	Circle SAL minus SCL for FC10 FC20 FC30 FC40 [40]	285
.3.27	Circle FC10 FrateCmd CurrFrate X-Frate Y-Frate [41]	285
.3.28	Circle FC20 FrateCmd CurrFrate X-Frate Y-Frate [42]	285
.3.29	Circle FC30 FrateCmd CurrFrate X-Frate Y-Frate [43]	285
.3.30	Circle FC40 FrateCmd CurrFrate X-Frate Y-Frate [44]	285

.3.31	Circle FC10 Four Components FeedrateLimit [45]	285
.3.32	Circle FC20 Four Components FeedrateLimit [46]	285
.3.33	Circle FC30 Four Components FeedrateLimit [47]	285
.3.34	Circle FC40 Four Components FeedrateLimit [48]	285
.3.35	Circle Histogram Points FC10 FC20 FC30 FC40 [49]	285
.3.36	Circle Table distribution of interpolated points [2]	285
.3.37	Circle Table FC10-20-30-40 Run Performance data [3]	285
.4	APPENDIX ELLIPSE CURVE	306
.4.1	Plot of Ellipse curve [50]	306
.4.2	Ellipse Radius of Curvature [51]	306
.4.3	Ellipse Validation in LinuxCNC [52]	306
.4.4	Ellipse Direction of Travel 3D [53]	306
.4.5	Ellipse First and Second Order Taylor's Approx [54]	306
.4.6	Ellipse First minus Second Order Taylor's Approx [55]	306
.4.7	Ellipse Separate First and Second Order Taylor's Approx [56]	306
.4.8	Ellipse Separation SAL and SCL [57]	306
.4.9	Ellipse Chord-error in close view 2 scales [58]	306
.4.10	Ellipse Four Components Feedrate Limit [59]	306
.4.11	Ellipse FrateCommand FrateLimit and Curr-Frate [60]	306
.4.12	Ellipse FeedRateLimit minus CurrFeedRate [61]	306
.4.13	Ellipse FC20-Nominal X and Y Feedrate Profiles [62]	306
.4.14	Ellipse FC20 Nominal Tangential Acceleration [63]	306
.4.15	Ellipse FC20 Nominal Rising S-Curve Profile [64]	306
.4.16	Ellipse FC20 Nominal Falling S-Curve Profile [65]	306
.4.17	Ellipse FC10 Colored Feedrate Profile data ngcode [66]	306
.4.18	Ellipse FC20 Colored Feedrate Profile data ngcode [67]	306
.4.19	Ellipse FC30 Colored Feedrate Profile data ngcode [68]	307
.4.20	Ellipse FC40 Colored Feedrate Profile data ngcode [69]	307
.4.21	Ellipse FC10 Tangential Acceleration [70]	307

.4.22	Ellipse FC20 Tangential Acceleration [71]	307
.4.23	Ellipse FC30 Tangential Acceleration [72]	307
.4.24	Ellipse FC40 Tangential Acceleration [73]	307
.4.25	Ellipse FC20 Nominal Separation NAL and NCL [74]	307
.4.26	Ellipse SAL minus SCL for FC10 FC20 FC30 FC40 [75]	307
.4.27	Ellipse FC10 FrateCmd CurrFrate X-Frate Y-Frate [76]	307
.4.28	Ellipse FC20 FrateCmd CurrFrate X-Frate Y-Frate [77]	307
.4.29	Ellipse FC30 FrateCmd CurrFrate X-Frate Y-Frate [78]	307
.4.30	Ellipse FC40 FrateCmd CurrFrate X-Frate Y-Frate [79]	307
.4.31	Ellipse FC10 Four Components FeedrateLimit [80]	307
.4.32	Ellipse FC20 Four Components FeedrateLimit [81]	307
.4.33	Ellipse FC30 Four Components FeedrateLimit [82]	307
.4.34	Ellipse FC40 Four Components FeedrateLimit [83]	307
.4.35	Ellipse Histogram Points FC10 FC20 FC30 FC40 [84]	307
.4.36	Ellipse Table distribution of interpolated points [4]	307
.4.37	Ellipse Table FC10-20-30-40 Run Performance data [5]	307
.5	APPENDIX TEARDROP CURVE	328
.5.1	Plot of Teardrop curve [85]	328
.5.2	Teardrop Radius of Curvature [86]	328
.5.3	Teardrop Validation in LinuxCNC [87]	328
.5.4	Teardrop Direction of Travel 3D [88]	328
.5.5	Teardrop First and Second Order Taylor's Approx [89]	328
.5.6	Teardrop First minus Second Order Taylor's Approx [90]	328
.5.7	Teardrop Separate First Second Order Taylor's Approx [91]	328
.5.8	Teardrop Separation SAL and SCL [92]	328
.5.9	Teardrop Chord-error in close view 2 scales [93]	328
.5.10	Teardrop Four Components Feedrate Limit [94]	328
.5.11	Teardrop FrateCommand FrateLimit and Curr-Frate [95]	328
.5.12	Teardrop FeedRateLimit minus CurrFeedRate [96]	328

.5.13	Teardrop FC20-Nominal X and Y Feedrate Profiles [97]	328
.5.14	Teardrop FC20 Nominal Tangential Acceleration [98]	328
.5.15	Teardrop FC20 Nominal Rising S-Curve Profile [99]	328
.5.16	Teardrop FC20 Nominal Falling S-Curve Profile [100]	328
.5.17	Teardrop FC10 Colored Feedrate Profile data ngcode [101]	328
.5.18	Teardrop FC20 Colored Feedrate Profile data ngcode [102]	328
.5.19	Teardrop FC30 Colored Feedrate Profile data ngcode [103]	329
.5.20	Teardrop FC40 Colored Feedrate Profile data ngcode [104]	329
.5.21	Teardrop FC10 Tangential Acceleration [105]	329
.5.22	Teardrop FC20 Tangential Acceleration [106]	329
.5.23	Teardrop FC30 Tangential Acceleration [107]	329
.5.24	Teardrop FC40 Tangential Acceleration [108]	329
.5.25	Teardrop FC20 Nominal Separation NAL and NCL [109]	329
.5.26	Teardrop SAL minus SCL for FC10 FC20 FC30 FC40 [110]	329
.5.27	Teardrop FC10 FrateCmd CurrFrate X-Frate Y-Frate [111]	329
.5.28	Teardrop FC20 FrateCmd CurrFrate X-Frate Y-Frate [112]	329
.5.29	Teardrop FC30 FrateCmd CurrFrate X-Frate Y-Frate [113]	329
.5.30	Teardrop FC40 FrateCmd CurrFrate X-Frate Y-Frate [114]	329
.5.31	Teardrop FC10 Four Components FeedrateLimit [115]	329
.5.32	Teardrop FC20 Four Components FeedrateLimit [116]	329
.5.33	Teardrop FC30 Four Components FeedrateLimit [117]	329
.5.34	Teardrop FC40 Four Components FeedrateLimit [118]	329
.5.35	Teardrop Histogram Points FC10 FC20 FC30 FC40 [119]	329
.5.36	Teardrop Table distribution of interpolated points [6]	329
.5.37	Teardrop Table FC10-20-30-40 Run Performance data [7]	329
.6	APPENDIX BUTTERFLY CURVE	350
.6.1	Plot of Butterfly curve [120]	350
.6.2	Butterfly Radius of Curvature [121]	350
.6.3	Butterfly Validation in LinuxCNC [122]	350

.6.4	Butterfly Direction of Travel 3D [123]	350
.6.5	Butterfly First and Second Order Taylor's Approx [124]	350
.6.6	Butterfly First minus Second Order Taylor's Approx [125]	350
.6.7	Butterfly Separate First Second Order Taylor's Approx [126]	350
.6.8	Butterfly Separation SAL and SCL [127]	350
.6.9	Butterfly Chord-error in close view 2 scales [128]	350
.6.10	Butterfly Four Components Feedrate Limit [129]	350
.6.11	Butterfly FrateCommand FrateLimit and Curr-Frate [130]	350
.6.12	Butterfly FeedRateLimit minus CurrFeedRate [131]	350
.6.13	Butterfly FC20-Nominal X and Y Feedrate Profiles [132]	350
.6.14	Butterfly FC20 Nominal Tangential Acceleration [133]	350
.6.15	Butterfly FC20 Nominal Rising S-Curve Profile [134]	350
.6.16	Butterfly FC20 Nominal Falling S-Curve Profile [135]	350
.6.17	Butterfly FC10 Colored Feedrate Profile data ngcode [136]	350
.6.18	Butterfly FC20 Colored Feedrate Profile data ngcode [137]	350
.6.19	Butterfly FC30 Colored Feedrate Profile data ngcode [138]	351
.6.20	Butterfly FC40 Colored Feedrate Profile data ngcode [139]	351
.6.21	Butterfly FC10 Tangential Acceleration [140]	351
.6.22	Butterfly FC20 Tangential Acceleration [141]	351
.6.23	Butterfly FC30 Tangential Acceleration [142]	351
.6.24	Butterfly FC40 Tangential Acceleration [143]	351
.6.25	Butterfly FC20 Nominal Separation NAL and NCL [144]	351
.6.26	Butterfly SAL minus SCL for FC10 FC20 FC30 FC40 [145]	351
.6.27	Butterfly FC10 FrateCmd CurrFrate X-Frate Y-Frate [146]	351
.6.28	Butterfly FC20 FrateCmd CurrFrate X-Frate Y-Frate [147]	351
.6.29	Butterfly FC30 FrateCmd CurrFrate X-Frate Y-Frate [148]	351
.6.30	Butterfly FC40 FrateCmd CurrFrate X-Frate Y-Frate [149]	351
.6.31	Butterfly FC10 Four Components FeedrateLimit [150]	351
.6.32	Butterfly FC20 Four Components FeedrateLimit [151]	351

.6.33	Butterfly FC30 Four Components FeedrateLimit [152]	351
.6.34	Butterfly FC40 Four Components FeedrateLimit [153]	351
.6.35	Butterfly Histogram Points FC10 FC20 FC30 FC40 [154]	351
.6.36	Butterfly Table distribution of interpolated points [8]	351
.6.37	Butterfly Table FC10-20-30-40 Run Performance data [9]	351
.7	APPENDIX SNAILSHELL CURVE	372
.7.1	Plot of Snailshell curve [155]	372
.7.2	Snailshell Radius of Curvature [156]	372
.7.3	Snailshell Validation in LinuxCNC [157]	372
.7.4	Snailshell Direction of Travel 3D [158]	372
.7.5	Snailshell First and Second Order Taylor's Approx [159]	372
.7.6	Snailshell First minus Second Order Taylor's Approx [160]	372
.7.7	Snailshell Separate First Second Order Taylor's Approx [161]	372
.7.8	Snailshell Separation SAL and SCL [162]	372
.7.9	Snailshell Chord-error in close view 2 scales [163]	372
.7.10	Snailshell Four Components Feedrate Limit [164]	372
.7.11	Snailshell FrateCommand FrateLimit and Curr-Frate [165]	372
.7.12	Snailshell FeedRateLimit minus CurrFeedRate [166]	372
.7.13	Snailshell FC20-Nominal X and Y Feedrate Profiles [167]	372
.7.14	Snailshell FC20 Nominal Tangential Acceleration [168]	372
.7.15	Snailshell FC20 Nominal Rising S-Curve Profile [169]	372
.7.16	Snailshell FC20 Nominal Falling S-Curve Profile [170]	372
.7.17	Snailshell FC10 Colored Feedrate Profile data ngcode [171]	372
.7.18	Snailshell FC20 Colored Feedrate Profile data ngcode [172]	372
.7.19	Snailshell FC30 Colored Feedrate Profile data ngcode [173]	373
.7.20	Snailshell FC40 Colored Feedrate Profile data ngcode [174]	373
.7.21	Snailshell FC10 Tangential Acceleration [175]	373
.7.22	Snailshell FC20 Tangential Acceleration [176]	373
.7.23	Snailshell FC30 Tangential Acceleration [177]	373

.7.24	Snailshell FC40 Tangential Acceleration [178]	373
.7.25	Snailshell FC20 Nominal Separation NAL and NCL [179]	373
.7.26	Snailshell SAL minus SCL for FC10 FC20 FC30 FC40 [180]	373
.7.27	Snailshell FC10 FrateCmd CurrFrate X-Frate Y-Frate [181]	373
.7.28	Snailshell FC20 FrateCmd CurrFrate X-Frate Y-Frate [182]	373
.7.29	Snailshell FC30 FrateCmd CurrFrate X-Frate Y-Frate [183]	373
.7.30	Snailshell FC40 FrateCmd CurrFrate X-Frate Y-Frate [184]	373
.7.31	Snailshell FC10 Four Components FeedrateLimit [185]	373
.7.32	Snailshell FC20 Four Components FeedrateLimit [186]	373
.7.33	Snailshell FC30 Four Components FeedrateLimit [187]	373
.7.34	Snailshell FC40 Four Components FeedrateLimit [188]	373
.7.35	Snailshell Histogram Points FC10 FC20 FC30 FC40 [189]	373
.7.36	Snailshell Table distribution of interpolated points [10]	373
.7.37	Snailshell Table FC10-20-30-40 Run Performance data [11]	373
.8	APPENDIX SKEWED-ASTROID CURVE	394
.8.1	Plot of SkwAstroid curve [190]	394
.8.2	SkwAstroid Radius of Curvature [191]	394
.8.3	SkwAstroid Validation in LinuxCNC [192]	394
.8.4	SkwAstroid Direction of Travel 3D [193]	394
.8.5	SkwAstroid First and Second Order Taylors App [194]	394
.8.6	SkwAstroid First minus Second Order Taylors App [195]	394
.8.7	SkwAstroid Separate First Second Order Taylors App [196]	394
.8.8	SkwAstroid Separation SAL and SCL [197]	394
.8.9	SkwAstroid Chord-error in close view 2 scales [198]	394
.8.10	SkwAstroid Four Components Feedrate Limit [199]	394
.8.11	SkwAstroid FrateCommand FrateLimit and Curr-Frate [200]	394
.8.12	SkwAstroid FeedRateLimit minus CurrFeedRate [201]	394
.8.13	SkwAstroid FC20-Nominal X and Y Feedrate Profiles [202]	394
.8.14	SkwAstroid FC20 Nominal Tangential Acceleration [203]	394

.8.15	SkwAstroid FC20 Nominal Rising S-Curve Profile [204]	394
.8.16	SkwAstroid FC20 Nominal Falling S-Curve Profile [205]	394
.8.17	SkwAstroid FC10 Colored Feedrate Profile ngcode [206]	394
.8.18	SkwAstroid FC20 Colored Feedrate Profile ngcode [207]	394
.8.19	SkwAstroid FC30 Colored Feedrate Profile ngcode [208]	395
.8.20	SkwAstroid FC40 Colored Feedrate Profile ngcode [209]	395
.8.21	SkwAstroid FC10 Tangential Acceleration [210]	395
.8.22	SkwAstroid FC20 Tangential Acceleration [211]	395
.8.23	SkwAstroid FC30 Tangential Acceleration [212]	395
.8.24	SkwAstroid FC40 Tangential Acceleration [213]	395
.8.25	SkwAstroid FC20 Nominal Separation NAL and NCL [214]	395
.8.26	SkwAstroid SAL minus SCL for FC10 FC20 FC30 FC40 [215]	395
.8.27	SkwAstroid FC10 FrateCmd CurrFrate X-Frate Y-Frate [216]	395
.8.28	SkwAstroid FC20 FrateCmd CurrFrate X-Frate Y-Frate [217]	395
.8.29	SkwAstroid FC30 FrateCmd CurrFrate X-Frate Y-Frate [218]	395
.8.30	SkwAstroid FC40 FrateCmd CurrFrate X-Frate Y-Frate [219]	395
.8.31	SkwAstroid FC10 Four Components FeedrateLimit [220]	395
.8.32	SkwAstroid FC20 Four Components FeedrateLimit [221]	395
.8.33	SkwAstroid FC30 Four Components FeedrateLimit [222]	395
.8.34	SkwAstroid FC40 Four Components FeedrateLimit [223]	395
.8.35	SkwAstroid Histogram Points FC10 FC20 FC30 FC40 [224]	395
.8.36	SkwAstroid Table distribution of interpolated points [12]	395
.8.37	SkwAstroid Table FC10-20-30-40 Run Performance data [13]	395
.9	APPENDIX RIBBON-10L CURVE	416
.9.1	Plot of Ribbon-10L curve [225]	416
.9.2	Ribbon-10L Radius of Curvature [226]	416
.9.3	Ribbon-10L Validation in LinuxCNC [227]	416
.9.4	Ribbon-10L Direction of Travel 3D [228]	416
.9.5	Ribbon-10L First and Second Order Taylors App [229]	416

.9.6	Ribbon-10L First minus Second Order Taylors App [230]	416
.9.7	Ribbon-10L Separate First Second Order Taylors App [??]	416
.9.8	Ribbon-10L Separation SAL and SCL [232]	416
.9.9	Ribbon-10L Chord-error in close view 2 scales [233]	416
.9.10	Ribbon-10L Four Components Feedrate Limit [234]	416
.9.11	Ribbon-10L FrateCommand FrateLimit and Curr-Frate [235]	416
.9.12	Ribbon-10L FeedRateLimit minus CurrFeedRate [236]	416
.9.13	Ribbon-10L FC20-Nominal X and Y Feedrate Profiles [237]	416
.9.14	Ribbon-10L FC20 Nominal Tangential Acceleration [238]	416
.9.15	Ribbon-10L FC20 Nominal Rising S-Curve Profile [239]	416
.9.16	Ribbon-10L FC20 Nominal Falling S-Curve Profile [240]	416
.9.17	Ribbon-10L FC10 Colored Feedrate Profile ngcode [241]	416
.9.18	Ribbon-10L FC20 Colored Feedrate Profile ngcode [242]	416
.9.19	Ribbon-10L FC30 Colored Feedrate Profile ngcode [243]	417
.9.20	Ribbon-10L FC40 Colored Feedrate Profile ngcode [244]	417
.9.21	Ribbon-10L FC10 Tangential Acceleration [245]	417
.9.22	Ribbon-10L FC20 Tangential Acceleration [246]	417
.9.23	Ribbon-10L FC30 Tangential Acceleration [247]	417
.9.24	Ribbon-10L FC40 Tangential Acceleration [248]	417
.9.25	Ribbon-10L FC20 Nominal Separation NAL and NCL [249]	417
.9.26	Ribbon-10L SAL minus SCL for FC10 FC20 FC30 FC40 [??]	417
.9.27	Ribbon-10L FC10 FrateCmd CurrFrate X-Frate Y-Frate [251]	417
.9.28	Ribbon-10L FC20 FrateCmd CurrFrate X-Frate Y-Frate [252]	417
.9.29	Ribbon-10L FC30 FrateCmd CurrFrate X-Frate Y-Frate [253]	417
.9.30	Ribbon-10L FC40 FrateCmd CurrFrate X-Frate Y-Frate [254]	417
.9.31	Ribbon-10L FC10 Four Components FeedrateLimit [255]	417
.9.32	Ribbon-10L FC20 Four Components FeedrateLimit [256]	417
.9.33	Ribbon-10L FC30 Four Components FeedrateLimit [257]	417
.9.34	Ribbon-10L FC40 Four Components FeedrateLimit [258]	417

.9.35	Ribbon-10L Histogram Points FC10 FC20 FC30 FC40 [259]	417
.9.36	Ribbon-10L Table distribution of interpolated points [14]	417
.9.37	Ribbon-10L Table FC10-20-30-40 Run Performance data [15]	417
.10	APPENDIX RIBBON-100L CURVE	438
.10.1	Plot of Ribbon-100L curve [260]	438
.10.2	Ribbon-100L Radius of Curvature [261]	438
.10.3	Ribbon-100L Validation in LinuxCNC [262]	438
.10.4	Ribbon-100L Direction of Travel 3D [263]	438
.10.5	Ribbon-100L First and Second Order Taylors App [264]	438
.10.6	Ribbon-100L First minus Second Order Taylors App [265]	438
.10.7	Ribbon-100L Separate First Second Order Taylors App [??]	438
.10.8	Ribbon-100L Separation SAL and SCL [267]	438
.10.9	Ribbon-100L Chord-error in close view 2 scales [268]	438
.10.10	Ribbon-100L Four Components Feedrate Limit [269]	438
.10.11	Ribbon-100L FrateCommand FrateLimit and Curr-Frate [270]	438
.10.12	Ribbon-100L FeedRateLimit minus CurrFeedRate [271]	438
.10.13	Ribbon-100L FC20-Nominal X and Y Feedrate Profiles [272]	438
.10.14	Ribbon-100L FC20 Nominal Tangential Acceleration [273]	438
.10.15	Ribbon-100L FC20 Nominal Rising S-Curve Profile [274]	438
.10.16	Ribbon-100L FC20 Nominal Falling S-Curve Profile [275]	438
.10.17	Ribbon-100L FC10 Colored Feedrate Profile ngcode [276]	438
.10.18	Ribbon-100L FC20 Colored Feedrate Profile ngcode [277]	438
.10.19	Ribbon-100L FC30 Colored Feedrate Profile ngcode [278]	439
.10.20	Ribbon-100L FC40 Colored Feedrate Profile ngcode [279]	439
.10.21	Ribbon-100L FC10 Tangential Acceleration [280]	439
.10.22	Ribbon-100L FC20 Tangential Acceleration [281]	439
.10.23	Ribbon-100L FC30 Tangential Acceleration [282]	439
.10.24	Ribbon-100L FC40 Tangential Acceleration [283]	439
.10.25	Ribbon-100L FC20 Nominal Separation NAL and NCL [284]	439

.10.26	Ribbon-100L SAL minus SCL FC10 FC20 FC30 FC40 [??]	439
.10.27	Ribbon-100L FC10 FrateCmd CurrFrate X-Frate Y-Frate [286]	439
.10.28	Ribbon-100L FC20 FrateCmd CurrFrate X-Frate Y-Frate [287]	439
.10.29	Ribbon-100L FC30 FrateCmd CurrFrate X-Frate Y-Frate [288]	439
.10.30	Ribbon-100L FC40 FrateCmd CurrFrate X-Frate Y-Frate [289]	439
.10.31	Ribbon-100L FC10 Four Components FeedrateLimit [290]	439
.10.32	Ribbon-100L FC20 Four Components FeedrateLimit [291]	439
.10.33	Ribbon-100L FC30 Four Components FeedrateLimit [292]	439
.10.34	Ribbon-100L FC40 Four Components FeedrateLimit [293]	439
.10.35	Ribbon-100L Histogram Points FC10 FC20 FC30 FC40 [294]	439
.10.36	Ribbon-100L Table distribution of interpolated points [16]	439
.10.37	Ribbon-100L Table FC10-20-30-40 Run Performance data [17]	439
.11	APPENDIX ASTEPI CURVE	460
.11.1	Plot of AstEpi curve [295]	460
.11.2	AstEpi Radius of Curvature [296]	460
.11.3	AstEpi Validation in LinuxCNC [297]	460
.11.4	AstEpi Direction of Travel 3D [298]	460
.11.5	AstEpi First and Second Order Taylors App [299]	460
.11.6	AstEpi First minus Second Order Taylors App [300]	460
.11.7	AstEpi Separate First Second Order Taylors App [301]	460
.11.8	AstEpi Separation SAL and SCL [302]	460
.11.9	AstEpi Chord-error in close view 2 scales [303]	460
.11.10	AstEpi Four Components Feedrate Limit [304]	460
.11.11	AstEpi FrateCommand FrateLimit and Curr-Frate [305]	460
.11.12	AstEpi FeedRateLimit minus CurrFeedRate [306]	460
.11.13	AstEpi FC20-Nominal X and Y Feedrate Profiles [307]	460
.11.14	AstEpi FC20 Nominal Tangential Acceleration [308]	460
.11.15	AstEpi FC20 Nominal Rising S-Curve Profile [309]	460
.11.16	AstEpi FC20 Nominal Falling S-Curve Profile [310]	460

.11.17	AstEpi FC10 Colored Feedrate Profile ngcode [311]	460
.11.18	AstEpi FC20 Colored Feedrate Profile ngcode [312]	460
.11.19	AstEpi FC30 Colored Feedrate Profile ngcode [313]	461
.11.20	AstEpi FC40 Colored Feedrate Profile ngcode [314]	461
.11.21	AstEpi FC10 Tangential Acceleration [315]	461
.11.22	AstEpi FC20 Tangential Acceleration [316]	461
.11.23	AstEpi FC30 Tangential Acceleration [317]	461
.11.24	AstEpi FC40 Tangential Acceleration [318]	461
.11.25	AstEpi FC20 Nominal Separation NAL and NCL [319]	461
.11.26	AstEpi SAL minus SCL for FC10 FC20 FC30 FC40 [320]	461
.11.27	AstEpi FC10 FrateCmd CurrFrate X-Frate Y-Frate [321]	461
.11.28	AstEpi FC20 FrateCmd CurrFrate X-Frate Y-Frate [322]	461
.11.29	AstEpi FC30 FrateCmd CurrFrate X-Frate Y-Frate [323]	461
.11.30	AstEpi FC40 FrateCmd CurrFrate X-Frate Y-Frate [324]	461
.11.31	AstEpi FC10 Four Components FeedrateLimit [325]	461
.11.32	AstEpi FC20 Four Components FeedrateLimit [326]	461
.11.33	AstEpi FC30 Four Components FeedrateLimit [327]	461
.11.34	AstEpi FC40 Four Components FeedrateLimit [328]	461
.11.35	AstEpi Histogram Points FC10 FC20 FC30 FC40 [329]	461
.11.36	AstEpi Table distribution of interpolated points [18]	461
.11.37	AstEpi Table FC10-20-30-40 Run Performance data [19]	461
.12	APPENDIX SNAHYP CURVE	482
.12.1	Plot of SnaHyp curve [330]	482
.12.2	SnaHyp Radius of Curvature [331]	482
.12.3	SnaHyp Validation in LinuxCNC [332]	482
.12.4	SnaHyp Direction of Travel 3D [333]	482
.12.5	SnaHyp First and Second Order Taylors App [334]	482
.12.6	SnaHyp First minus Second Order Taylors App [335]	482
.12.7	SnaHyp Separate First Second Order Taylors App [336]	482

.12.8 SnaHyp Separation SAL and SCL [337]	482
.12.9 SnaHyp Chord-error in close view 2 scales [338]	482
.12.10 SnaHyp Four Components Feedrate Limit [339]	482
.12.11 SnaHyp FrateCommand FrateLimit and Curr-Frate [340]	482
.12.12 SnaHyp FeedRateLimit minus CurrFeedRate [341]	482
.12.13 SnaHyp FC20-Nominal X and Y Feedrate Profiles [342]	482
.12.14 SnaHyp FC20 Nominal Tangential Acceleration [343]	482
.12.15 SnaHyp FC20 Nominal Rising S-Curve Profile [344]	482
.12.16 SnaHyp FC20 Nominal Falling S-Curve Profile [345]	482
.12.17 SnaHyp FC10 Colored Feedrate Profile ngcode [346]	482
.12.18 SnaHyp FC20 Colored Feedrate Profile ngcode [347]	482
.12.19 SnaHyp FC25 Colored Feedrate Profile ngcode [348]	483
.12.20 SnaHyp FC28 Colored Feedrate Profile ngcode [349]	483
.12.21 SnaHyp FC10 Tangential Acceleration [350]	483
.12.22 SnaHyp FC20 Tangential Acceleration [351]	483
.12.23 SnaHyp FC30 Tangential Acceleration [352]	483
.12.24 SnaHyp FC40 Tangential Acceleration [353]	483
.12.25 SnaHyp FC20 Nominal Separation NAL and NCL [354]	483
.12.26 SnaHyp SAL minus SCL for FC10 FC20 FC30 FC40 [355]	483
.12.27 SnaHyp FC10 FrateCmd CurrFrate X-Frate Y-Frate [356]	483
.12.28 SnaHyp FC20 FrateCmd CurrFrate X-Frate Y-Frate [357]	483
.12.29 SnaHyp FC25 FrateCmd CurrFrate X-Frate Y-Frate [358]	483
.12.30 SnaHyp FC28 FrateCmd CurrFrate X-Frate Y-Frate [359]	483
.12.31 SnaHyp FC10 Four Components FeedrateLimit [360]	483
.12.32 SnaHyp FC20 Four Components FeedrateLimit [361]	483
.12.33 SnaHyp FC25 Four Components FeedrateLimit [362]	483
.12.34 SnaHyp FC28 Four Components FeedrateLimit [363]	483
.12.35 SnaHyp Histogram Points FC10 FC20 FC30 FC40 [364]	483
.12.36 SnaHyp Table distribution of interpolated points [20]	483

LIST OF TABLES

Table 3.1:	List of ten(10) selected parametric curves	30
Table 3.2:	Teardrop curve parametric equation	36
Table 3.3:	Butterfly curve parametric equation	38
Table 3.4:	Ellipse curve parametric equation	40
Table 3.5:	SkewedAstroid curve parametric equation	41
Table 3.6:	Circle parametric equation	42
Table 3.7:	AstEpi curve parametric equation	43
Table 3.8:	Snailshell curve parametric equation	44
Table 3.9:	SnaHyp curve parametric equation	45
Table 3.10:	Ribbon10L curve parametric equation	46
Table 3.11:	Ribbon100L curve parametric equation	47
Table 3.12:	Feedrate limit function parameters	62
Table 3.13:	Calculation for fratelimit_1	63
Table 3.14:	Calculation for fratelimit_2	63
Table 3.15:	Calculation for fratelimit_3	64
Table 3.16:	Calculation for fratelimit_4	65
Table 3.17:	Rising S-curve parameters	66
Table 3.18:	Falling S-curve parameters	80
Table 3.19:	Acceleration safety factor (lambda)	81
Table 3.20:	Definitions of SAL(u), SCL(u), SAT(u), SAA(u) and SCE(u)	83
Table 3.21:	Definitions of Total SAL, Total SCL, Total SAT, Total SAA, and Total SCE	83
Table 3.22:	Algorithm and runtime parameters	95
Table 3.23:	System Hardware Specifications	98
Table 3.24:	Operating Systems	99
Table 3.25:	Programming Software Languages	99

Table 3.26: Reporting Software Applications	100
Table 3.27: Specialized Software Applications	100
 Table 4.1: Results for finding acceptable lamda	131
Table 4.2: Results for rechecking acceptable lamda	132
Table 4.3: Teardrop Table distribution of interpolated points	164
Table 4.4: Terms used in Curve Performance data	173
Table 4.5: Teardrop Table FC10-20-30-40 Run Performance data	177
Table 4.6: Teardrop SCE/TIP Table FC10-20-30-40 Run Performance data	179
Table 4.7: Teardrop SCE/SCL Table FC10-20-30-40 Run Performance data	180
Table 4.8: Teardrop SAA/SCL Table FC10-20-30-40 Run Performance data	181
Table 4.9: Teardrop (SAL-SCL)/SAL Table FC10-20-30-40 Run Performance data	182
Table 4.10: Circle Table FC10-20-30-40 Run Performance data	190
Table 4.11: Ellipse Table FC10-20-30-40 Run Performance data	191
Table 4.12: Butterfly Table FC10-20-30-40 Run Performance data	192
Table 4.13: Snailshell Table FC10-20-30-40 Run Performance data	193
Table 4.14: Skewed-Astroid Table FC10-20-30-40 Run Performance data	194
Table 4.15: Ribbon-10L Table FC10-20-30-40 Run Performance data	195
Table 4.16: Ribbon-100L Table FC10-20-30-40 Run Performance data	196
Table 4.17: AstEpi Table FC10-20-30-40 Run Performance data	197
Table 4.18: SnaHyp Table FC10-20-30-40 Run Performance data	198
Table 4.19: Circle - Algorithm Validation and Verification	199
Table 4.20: Ellipse - Algorithm Validation and Verification	200
Table 4.21: Anomaly at FC30 Teardrop Table FC10-20-30-40 Run Performance data	204
Table 4.22: Overall Total Interpolated Points Data TIP	223
Table 4.23: Overall Total Sum-Arc-Length Data SAL	226
Table 4.24: Overall Total Sum-Chord-Length SCL	229
Table 4.25: Overall Percentage Difference between SAL and SCL	233

Table 4.26:	Overall Sum-Chord-Error Data SCE	236
Table 4.27:	Overall Performance 2 Chord-Error-Efficiency CEE	239
Table 4.28:	Overall Sum-Arc-Area SAA	243
Table 4.29:	Overall Arc-Area-Efficiency AAE	246
Table 5.1:	Chord-error per unit length curve traversed (ratio) CEE	251
Table 2:	Circle Table distribution of interpolated points	303
Table 3:	Circle Table FC10-20-30-40 Run Performance data	304
Table 4:	Ellipse Table distribution of interpolated points	325
Table 5:	Ellipse Table FC10-20-30-40 Run Performance data	326
Table 6:	Teardrop Table distribution of interpolated points	347
Table 7:	Teardrop Table FC10-20-30-40 Run Performance data	348
Table 8:	Butterfly Table distribution of interpolated points	369
Table 9:	Butterfly Table FC10-20-30-40 Run Performance data	370
Table 10:	Snailshell Table distribution of interpolated points	391
Table 11:	Snailshell Table FC10-20-30-40 Run Performance data	392
Table 12:	SkwAstroid Table distribution of interpolated points	413
Table 13:	SkwAstroid Table FC10-20-30-40 Run Performance data	414
Table 14:	Ribbon-10L Table distribution of interpolated points	435
Table 15:	Ribbon-10L Table FC10-20-30-40 Run Performance data	436
Table 16:	Ribbon-100L Table distribution of interpolated points	457
Table 17:	Ribbon-100L Table FC10-20-30-40 Run Performance data	458
Table 18:	AstEpi Table distribution of interpolated points	479
Table 19:	AstEpi Table FC10-20-30-40 Run Performance data	480
Table 20:	SnaHyp Table distribution of interpolated points	501
Table 21:	SnaHyp Table FC10-20-30-40 Run Performance data	502

LIST OF FIGURES

Figure 1.1:	Concept of Chord-error ϵ , Feedrate F, and Interpolation time T	2
Figure 1.2:	Butterfly Perspective View 3D in LinuxCNC-Axis	6
Figure 2.1:	Basic architecture for a typical CNC system	13
Figure 2.2:	Typical Servo drive and spindle driving mechanism	14
Figure 2.3:	Typical CNC Software control flow diagram	14
Figure 2.4:	Typical CNC Functional Architecture	15
Figure 2.5:	Typical CNC Commercial Milling machine	16
Figure 2.6:	Typical CNC Commercial Lathe machine	17
Figure 2.7:	CNC UMP Model 2008 - University Malaysia Pahang	24
Figure 2.8:	Youtube: CNC Jalur Gemilang Rendition Video by Ahmad (2017)	24
Figure 2.9:	Proton logo output of UMP CNC machine by Ahmad:2017	25
Figure 2.10:	Setup of Nexsys-3 Spartan-6 Xilinx board by Teh:2016	25
Figure 2.11:	G-code file (left) and Signal file (right) for FPGA by Teh:2016	29
Figure 3.1:	Table data summary curve shapes and dimensions	34
Figure 3.2:	Teardrop shape profile	36
Figure 3.3:	Teardrop comparison Zhong et. al. (2018)	37
Figure 3.4:	Butterfly shape profile	38
Figure 3.5:	Butterfly comparison Ni et. al. (2018) NURBS	39
Figure 3.6:	Butterfly comparison Hu et. al. (2023) NURBS	39
Figure 3.7:	Ellipse shape profile	40
Figure 3.8:	SkewedAstroid shape profile	41
Figure 3.9:	Circle shape profile	42
Figure 3.10:	AstEpi shape profile	43
Figure 3.11:	SnaIshell shape profile	44
Figure 3.12:	SnaHyp shape profile	45

Figure 3.13: Ribbon10L shape profile	46
Figure 3.14: Ribbon100L shape profile	47
Figure 3.15: Ribbon comparison Zhong et. al. (2018) B-Splines	48
Figure 3.16: Teardrop 3D plot x(u) vs y(u) vs u(t)	49
Figure 3.17: Butterfly 3D plot x(u) vs y(u) vs u(t)	50
Figure 3.18: Chord-error ϵ , Feedrate F, and Interpolation time T	51
Figure 3.19: Example-1-Structured Programming Design (1-in, 1-out)	53
Figure 3.20: Example-2-Structured Programming Design (1-in, 1-out)	53
Figure 3.21: Example-3-Non-Structured Programming Design, (S=Start, E=Exit)	54
Figure 3.22: Example-4-Non-Structured Programming Design (1-in, 4-outs)	54
Figure 3.23: Chord-error between two interpolated points	56
Figure 3.24: Flowchart of main program	69
Figure 3.25: Flowchart of Feedrate Rising S-Curve	71
Figure 3.26: Flowchart Current Feedrate above Feedrate Limit Part 1/2	72
Figure 3.27: Flowchart Current Feedrate above Feedrate Limit Part 2/2	73
Figure 3.28: Flowchart Current Feedrate below Feedrate Limit Part 1/2	74
Figure 3.29: Flowchart Current Feedrate below Feedrate Limit Part 2/2	75
Figure 3.30: Flowchart Current Feedrate equal Feedrate Limit	76
Figure 3.31: Flowchart Calculate $u_{\text{next}}(u)$	77
Figure 3.32: Flowchart Calculate $u_{\text{next}}(u)$ with ϵ pushdown	78
Figure 3.33: Flowchart of Feedrate Falling S-Curve	79
Figure 3.34: Calculation for the Circle Arc-Length	84
Figure 3.35: Algorithm-Functional-Components	93
Figure 3.36: The prototype 3-axis CNC research machine	102
Figure 3.37: The 3-sets AC servo drivers for the CNC research machine	102
Figure 3.38: Validation of G-code using LinuxCNC Axis software	103
Figure 3.39: The front end environment of the system	105
Figure 3.40: Overview CNC system menvironment	106
Figure 3.41: CNC-system-Panaterm-controller	107

Figure 3.42: Parport-CNC-servo-PCIE-1884-wiring-diagram	108
Figure 3.43: RS232 Snooper Circuit	109
Figure 4.1: Feedrate variations along the curve	116
Figure 4.2: Comparisons velocity profiles of the x and y axes motions	117
Figure 4.3: Running feedrate profile against the feedrate-limit	118
Figure 4.4: Acceleration profile containment	119
Figure 4.5: Execution of Teardrop Curve on CNC Machine	121
Figure 4.6: Results of Teardrop Curve CNC Machine execution - ruler (cm)	121
Figure 4.7: Teardrop Perspective View 3D in LinuxCNC-Axis	123
Figure 4.8: Example Snailshell Tangential Acceleration Jitters	127
Figure 4.9: Example Snailshell Close View Tangential Acceleration Jitters	128
Figure 4.10: Example Ribbon-100L Tangential Acceleration Jitters	129
Figure 4.11: Example Ribbon-100L Close View Tangential Acceleration Jitters	130
Figure 4.12: Circle Lamda Safety Factor Threshold	133
Figure 4.13: Ellipse Lamda Safety Factor Threshold	134
Figure 4.14: Teardrop Lamda Safety Factor Threshold	135
Figure 4.15: Butterfly Lamda Safety Factor Threshold	136
Figure 4.16: Snailshell Lamda Safety Factor Threshold	137
Figure 4.17: Skewed-Astroid Lamda Safety Factor Threshold	138
Figure 4.18: Ribbon-10L Lamda Safety Factor Threshold	139
Figure 4.19: Ribbon-100L Lamda Safety Factor Threshold	140
Figure 4.20: AstEpi Lamda Safety Factor Threshold	141
Figure 4.21: SnaHyp Lamda Safety Factor Threshold	142
Figure 4.22: Butterfly Tangential Acceleration Magnified 2X	143
Figure 4.23: Butterfly Tangential Acceleration Closeup View	144
Figure 4.24: SnaHyp Tangential Acceleration Magnified 2X	145
Figure 4.25: SnaHyp Tangential Acceleration Closeup View	146
Figure 4.26: Plot of Teardrop curve	152
Figure 4.27: Teardrop Direction of Travel 3D	152

Figure 4.28: Teardrop Perspective View 3D in LinuxCNC-Axis	153
Figure 4.29: Teardrop Radius of Curvature	155
Figure 4.30: Butterfly Radius of Curvature	155
Figure 4.31: Teardrop 1st and 2nd Order Taylor's Approximation	157
Figure 4.32: Teardrop (1st minus 2nd) Order Taylor's Approximation	157
Figure 4.33: Teardrop Chord-Error Absolute Constraint	160
Figure 4.34: Teardrop Four Feedrate Limit Components Profile	160
Figure 4.35: Teardrop FRate Command, FRate Limit and Current FRate	162
Figure 4.36: Teardrop Current Feedrate Absolute Constraint	162
Figure 4.37: Teardrop Histogram distribution of interpolated points	164
Figure 4.38: Teardrop Rising Current Feedrate Profile	166
Figure 4.39: Teardrop Falling Current Feedrate Profile	166
Figure 4.40: Teardrop FC10 Running Feedrate Profiles	168
Figure 4.41: Teardrop FC20 Running Feedrate Profiles	168
Figure 4.42: Teardrop FC30 Running Feedrate Profiles	169
Figure 4.43: Teardrop FC40 Running Feedrate Profiles	169
Figure 4.44: Teardrop FC10 Colored Feedrate Run Profile	171
Figure 4.45: Teardrop FC20 Colored Feedrate Run Profile	171
Figure 4.46: Teardrop FC30 Colored Feedrate Run Profile	172
Figure 4.47: Teardrop FC40 Colored Feedrate Run Profile	172
Figure 4.48: Teardrop Run Performance 1: SCE/TIP	179
Figure 4.49: Teardrop Run Performance 2: SCE/SCL	180
Figure 4.50: Teardrop Run Performance 3: SAA/SCL	181
Figure 4.51: Teardrop Run Performance 4: 100*(SAL-SCL)/SAL	182
Figure 4.52: Teardrop Closeup-View No Jitters FC40 Tangential Acceleration Run Profile	184
Figure 4.53: Teardrop FC10 Tangential Acceleration Run Profile	185
Figure 4.54: Teardrop FC20 Tangential Acceleration Run Profile	186
Figure 4.55: Teardrop FC30 Tangential Acceleration Run Profile	187

Figure 4.56: Teardrop FC40 Tangential Acceleration Run Profile	188
Figure 4.57: Plot of Teardrop FC10 FC20 FC30 and FC40 Differences SAL-SCL	206
Figure 4.58: Teardrop Separation of FC30 and FC40 Difference SAL-SCL	206
Figure 4.59: Plot of Teardrop Overlap FC30 and FC40 Differences SAL-SCL	207
Figure 4.60: Teardrop Separation Closeup of FC30 and FC40 Difference SAL-SCL	207
Figure 4.61: Plot of Teardrop FC10 NAL and NCL	208
Figure 4.62: Plot of Teardrop FC20 NAL and NCL	208
Figure 4.63: Plot of Teardrop FC30 NAL and NCL	209
Figure 4.64: Plot of Teardrop FC40 NAL and NCL	209
Figure 4.65: Histogram Overall Total Interpolated Points TIP	222
Figure 4.66: Histogram Overall Total Sum-Arc-Length SAL	225
Figure 4.67: Histogram Overall Total Sum-Chord-Length SCL	228
Figure 4.68: Histogram Linear Overall Percentage Difference between SAL and SCL	231
Figure 4.69: Histogram Log10 Overall Percentage Difference between SAL and SCL	232
Figure 4.70: Histogram Overall Sum-Chord-Error SCE	235
Figure 4.71: Histogram Overall Chord-Error-Efficiency CEE	238
Figure 4.72: Histogram Overall Sum-Arc-Area SAA	241
Figure 4.73: Histogram Log 10 Overall Sum-Arc-Area SAA	242
Figure 4.74: Histogram Log 10 Overall Arc-Area-Efficiency AAE	245
 Figure 1: Histogram-Latency-HP-Laptop-01-Debian10-10million-cycles	263
Figure 2: Histogram-Latency-HP-Laptop-01-Ubuntu20-10million-cycles	263
Figure 3: Histogram-Latency-HP-Laptop-02-Debian10-10million-cycles	264
Figure 4: Histogram-Latency-HP-Laptop-02-Ubuntu20-10million-cycles	264
Figure 5: Circle validation LinuxCNC-Axis execution	273
Figure 6: Ellipse validation LinuxCNC-Axis execution	274
Figure 7: Teardrop validation LinuxCNC-Axis execution	275

Figure 8:	Butterfly validation LinuxCNC-Axis execution	276
Figure 9:	Snailshell validation LinuxCNC-Axis execution	277
Figure 10:	Skewed-Astroid validation LinuxCNC-Axis execution	278
Figure 11:	Ribbon-10L validation LinuxCNC-Axis execution	279
Figure 12:	Ribbon-100L validation LinuxCNC-Axis execution	280
Figure 13:	AstEpi validation LinuxCNC-Axis execution	281
Figure 14:	SnaHyp validation LinuxCNC-Axis execution	282
Figure 15:	Plot of Circle curve	286
Figure 16:	Circle Radius of Curvature	286
Figure 17:	Circle Validation in LinuxCNC	287
Figure 18:	Circle Direction of Travel 3D	287
Figure 19:	Circle First and Second Order Taylor's Approximation	288
Figure 20:	Circle First minus Second Order Taylor's Approximation	288
Figure 21:	Circle Separation First and Second Order Taylor's Approximation	289
Figure 22:	Circle Separation SAL and SCL	289
Figure 23:	Circle Chord-error in close view 2 scales	290
Figure 24:	Circle Four Components Feedrate Limit	290
Figure 25:	Circle FrateCommand FrateLimit and Curr-Frate	291
Figure 26:	Circle FeedRateLimit minus CurrFeedRate	291
Figure 27:	Circle FC20-Nominal X and Y Feedrate Profiles	292
Figure 28:	Circle FC20 Nominal Tangential Acceleration	292
Figure 29:	Circle FC20 Nominal Rising S-Curve Profile	293
Figure 30:	Circle FC20 Nominal Falling S-Curve Profile	293
Figure 31:	Circle FC10 Colored Feedrate Profile data ngcode	294
Figure 32:	Circle FC20 Colored Feedrate Profile data ngcode	294
Figure 33:	Circle FC30 Colored Feedrate Profile data ngcode	295
Figure 34:	Circle FC40 Colored Feedrate Profile data ngcode	295
Figure 35:	Circle FC10 Tangential Acceleration	296
Figure 36:	Circle FC20 Tangential Acceleration	296

Figure 37:	Circle FC30 Tangential Acceleration	297
Figure 38:	Circle FC40 Tangential Acceleration	297
Figure 39:	Circle FC20 Nominal Separation NAL and NCL	298
Figure 40:	Circle Difference SAL minus SCL for FC10 FC20 FC30 FC40	298
Figure 41:	Circle FC10 FrateCmd CurrFrate X-Frate Y-Frate	299
Figure 42:	Circle FC20 FrateCmd CurrFrate X-Frate Y-Frate	299
Figure 43:	Circle FC30 FrateCmd CurrFrate X-Frate Y-Frate	300
Figure 44:	Circle FC40 FrateCmd CurrFrate X-Frate Y-Frate	300
Figure 45:	Circle FC10 Four Components FeedrateLimit	301
Figure 46:	Circle FC20 Four Components FeedrateLimit	301
Figure 47:	Circle FC30 Four Components FeedrateLimit	302
Figure 48:	Circle FC40 Four Components FeedrateLimit	302
Figure 49:	Circle Histogram Points FC10 FC20 FC30 FC40	303
Figure 50:	Plot of Ellipse curve	308
Figure 51:	Ellipse Radius of Curvature	308
Figure 52:	Ellipse Validation in LinuxCNC	309
Figure 53:	Ellipse Direction of Travel 3D	309
Figure 54:	Ellipse First and Second Order Taylor's Approximation	310
Figure 55:	Ellipse First minus Second Order Taylor's Approximation	310
Figure 56:	Ellipse Separation First and Second Order Taylor's Approximation	311
Figure 57:	Ellipse Separation SAL and SCL	311
Figure 58:	Ellipse Chord-error in close view 2 scales	312
Figure 59:	Ellipse Four Components Feedrate Limit	312
Figure 60:	Ellipse FrateCommand FrateLimit and Curr-Frate	313
Figure 61:	Ellipse FeedRateLimit minus CurrFeedRate	313
Figure 62:	Ellipse FC20-Nominal X and Y Feedrate Profiles	314
Figure 63:	Ellipse FC20 Nominal Tangential Acceleration	314
Figure 64:	Ellipse FC20 Nominal Rising S-Curve Profile	315
Figure 65:	Ellipse FC20 Nominal Falling S-Curve Profile	315

Figure 66:	Ellipse FC10 Colored Feedrate Profile data ngcode	316
Figure 67:	Ellipse FC20 Colored Feedrate Profile data ngcode	316
Figure 68:	Ellipse FC30 Colored Feedrate Profile data ngcode	317
Figure 69:	Ellipse FC40 Colored Feedrate Profile data ngcode	317
Figure 70:	Ellipse FC10 Tangential Acceleration	318
Figure 71:	Ellipse FC20 Tangential Acceleration	318
Figure 72:	Ellipse FC30 Tangential Acceleration	319
Figure 73:	Ellipse FC40 Tangential Acceleration	319
Figure 74:	Ellipse FC20 Nominal Separation NAL and NCL	320
Figure 75:	Ellipse Difference SAL minus SCL for FC10 FC20 FC30 FC40	320
Figure 76:	Ellipse FC10 FrateCmd CurrFrate X-Frate Y-Frate	321
Figure 77:	Ellipse FC20 FrateCmd CurrFrate X-Frate Y-Frate	321
Figure 78:	Ellipse FC30 FrateCmd CurrFrate X-Frate Y-Frate	322
Figure 79:	Ellipse FC40 FrateCmd CurrFrate X-Frate Y-Frate	322
Figure 80:	Ellipse FC10 Four Components FeedrateLimit	323
Figure 81:	Ellipse FC20 Four Components FeedrateLimit	323
Figure 82:	Ellipse FC30 Four Components FeedrateLimit	324
Figure 83:	Ellipse FC40 Four Components FeedrateLimit	324
Figure 84:	Ellipse Histogram Points FC10 FC20 FC30 FC40	325
Figure 85:	Plot of Teardrop curve	330
Figure 86:	Teardrop Radius of Curvature	330
Figure 87:	Teardrop Validation in LinuxCNC	331
Figure 88:	Teardrop Direction of Travel 3D	331
Figure 89:	Teardrop First and Second Order Taylor's Approximation	332
Figure 90:	Teardrop First minus Second Order Taylor's Approximation	332
Figure 91:	Teardrop Separation First and Second Order Taylor's Approximation	333
Figure 92:	Teardrop Separation SAL and SCL	333
Figure 93:	Teardrop Chord-error in close view 2 scales	334

Figure 94:	Teardrop Four Components Feedrate Limit	334
Figure 95:	Teardrop FrateCommand FrateLimit and Curr-Frate	335
Figure 96:	Teardrop FeedRateLimit minus CurrFeedRate	335
Figure 97:	Teardrop FC20-Nominal X and Y Feedrate Profiles	336
Figure 98:	Teardrop FC20 Nominal Tangential Acceleration	336
Figure 99:	Teardrop FC20 Nominal Rising S-Curve Profile	337
Figure 100:	Teardrop FC20 Nominal Falling S-Curve Profile	337
Figure 101:	Teardrop FC10 Colored Feedrate Profile data ngcode	338
Figure 102:	Teardrop FC20 Colored Feedrate Profile data ngcode	338
Figure 103:	Teardrop FC30 Colored Feedrate Profile data ngcode	339
Figure 104:	Teardrop FC40 Colored Feedrate Profile data ngcode	339
Figure 105:	Teardrop FC10 Tangential Acceleration	340
Figure 106:	Teardrop FC20 Tangential Acceleration	340
Figure 107:	Teardrop FC30 Tangential Acceleration	341
Figure 108:	Teardrop FC40 Tangential Acceleration	341
Figure 109:	Teardrop FC20 Nominal Separation NAL and NCL	342
Figure 110:	Teardrop Difference SAL minus SCL for FC10 FC20 FC30 FC40	342
Figure 111:	Teardrop FC10 FrateCmd CurrFrate X-Frate Y-Frate	343
Figure 112:	Teardrop FC20 FrateCmd CurrFrate X-Frate Y-Frate	343
Figure 113:	Teardrop FC30 FrateCmd CurrFrate X-Frate Y-Frate	344
Figure 114:	Teardrop FC40 FrateCmd CurrFrate X-Frate Y-Frate	344
Figure 115:	Teardrop FC10 Four Components FeedrateLimit	345
Figure 116:	Teardrop FC20 Four Components FeedrateLimit	345
Figure 117:	Teardrop FC30 Four Components FeedrateLimit	346
Figure 118:	Teardrop FC40 Four Components FeedrateLimit	346
Figure 119:	Teardrop Histogram Points FC10 FC20 FC30 FC40	347
Figure 120:	Plot of Butterfly curve	352
Figure 121:	Butterfly Radius of Curvature	352
Figure 122:	Butterfly Validation in LinuxCNC	353

Figure 123: Butterfly Direction of Travel 3D	353
Figure 124: Butterfly First and Second Order Taylor's Approximation	354
Figure 125: Butterfly First minus Second Order Taylor's Approximation	354
Figure 126: Butterfly Separation First and Second Order Taylor's Approximation	355
Figure 127: Butterfly Separation SAL and SCL	355
Figure 128: Butterfly Chord-error in close view 2 scales	356
Figure 129: Butterfly Four Components Feedrate Limit	356
Figure 130: Butterfly FrateCommand FrateLimit and Curr-Frate	357
Figure 131: Butterfly FeedRateLimit minus CurrFeedRate	357
Figure 132: Butterfly FC20-Nominal X and Y Feedrate Profiles	358
Figure 133: Butterfly FC20 Nominal Tangential Acceleration	358
Figure 134: Butterfly FC20 Nominal Rising S-Curve Profile	359
Figure 135: Butterfly FC20 Nominal Falling S-Curve Profile	359
Figure 136: Butterfly FC10 Colored Feedrate Profile data ngcode	360
Figure 137: Butterfly FC20 Colored Feedrate Profile data ngcode	360
Figure 138: Butterfly FC30 Colored Feedrate Profile data ngcode	361
Figure 139: Butterfly FC40 Colored Feedrate Profile data ngcode	361
Figure 140: Butterfly FC10 Tangential Acceleration	362
Figure 141: Butterfly FC20 Tangential Acceleration	362
Figure 142: Butterfly FC30 Tangential Acceleration	363
Figure 143: Butterfly FC40 Tangential Acceleration	363
Figure 144: Butterfly FC20 Nominal Separation NAL and NCL	364
Figure 145: Butterfly Difference SAL minus SCL for FC10 FC20 FC30 FC40	364
Figure 146: Butterfly FC10 FrateCmd CurrFrate X-Frate Y-Frate	365
Figure 147: Butterfly FC20 FrateCmd CurrFrate X-Frate Y-Frate	365
Figure 148: Butterfly FC30 FrateCmd CurrFrate X-Frate Y-Frate	366
Figure 149: Butterfly FC40 FrateCmd CurrFrate X-Frate Y-Frate	366
Figure 150: Butterfly FC10 Four Components FeedrateLimit	367

Figure 151: Butterfly FC20 Four Components FeedrateLimit	367
Figure 152: Butterfly FC30 Four Components FeedrateLimit	368
Figure 153: Butterfly FC40 Four Components FeedrateLimit	368
Figure 154: Butterfly Histogram Points FC10 FC20 FC30 FC40	369
Figure 155: Plot of Snailshell curve	374
Figure 156: Snailshell Radius of Curvature	374
Figure 157: Snailshell Validation in LinuxCNC	375
Figure 158: Snailshell Direction of Travel 3D	375
Figure 159: Snailshell First and Second Order Taylor's Approximation	376
Figure 160: Snailshell First minus Second Order Taylor's Approximation	376
Figure 161: Snailshell Separation First and Second Order Taylor's Approximation	377
Figure 162: Snailshell Separation SAL and SCL	377
Figure 163: Snailshell Chord-error in close view 2 scales	378
Figure 164: Snailshell Four Components Feedrate Limit	378
Figure 165: Snailshell FrateCommand FrateLimit and Curr-Frate	379
Figure 166: Snailshell FeedRateLimit minus CurrFeedRate	379
Figure 167: Snailshell FC20-Nominal X and Y Feedrate Profiles	380
Figure 168: Snailshell FC20 Nominal Tangential Acceleration	380
Figure 169: Snailshell FC20 Nominal Rising S-Curve Profile	381
Figure 170: Snailshell FC20 Nominal Falling S-Curve Profile	381
Figure 171: Snailshell FC10 Colored Feedrate Profile data ngcode	382
Figure 172: Snailshell FC20 Colored Feedrate Profile data ngcode	382
Figure 173: Snailshell FC30 Colored Feedrate Profile data ngcode	383
Figure 174: Snailshell FC40 Colored Feedrate Profile data ngcode	383
Figure 175: Snailshell FC10 Tangential Acceleration	384
Figure 176: Snailshell FC20 Tangential Acceleration	384
Figure 177: Snailshell FC30 Tangential Acceleration	385
Figure 178: Snailshell FC40 Tangential Acceleration	385

Figure 179: Snailshell FC20 Nominal Separation NAL and NCL	386
Figure 180: Snailshell Difference SAL minus SCL for FC10 FC20 FC30 FC40	386
Figure 181: Snailshell FC10 FrateCmd CurrFrate X-Frate Y-Frate	387
Figure 182: Snailshell FC20 FrateCmd CurrFrate X-Frate Y-Frate	387
Figure 183: Snailshell FC30 FrateCmd CurrFrate X-Frate Y-Frate	388
Figure 184: Snailshell FC40 FrateCmd CurrFrate X-Frate Y-Frate	388
Figure 185: Snailshell FC10 Four Components FeedrateLimit	389
Figure 186: Snailshell FC20 Four Components FeedrateLimit	389
Figure 187: Snailshell FC30 Four Components FeedrateLimit	390
Figure 188: Snailshell FC40 Four Components FeedrateLimit	390
Figure 189: Snailshell Histogram Points FC10 FC20 FC30 FC40	391
Figure 190: Plot of SkwAstroid curve	396
Figure 191: SkwAstroid Radius of Curvature	396
Figure 192: SkwAstroid Validation in LinuxCNC	397
Figure 193: SkwAstroid Direction of Travel 3D	397
Figure 194: SkwAstroid First and Second Order Taylor's Approximation	398
Figure 195: SkwAstroid First minus Second Order Taylor's Approximation	398
Figure 196: SkwAstroid Separation First and Second Order Taylor's Approximation	399
Figure 197: SkwAstroid Separation SAL and SCL	399
Figure 198: SkwAstroid Chord-error in close view 2 scales	400
Figure 199: SkwAstroid Four Components Feedrate Limit	400
Figure 200: SkwAstroid FrateCommand FrateLimit and Curr-Frate	401
Figure 201: SkwAstroid FeedRateLimit minus CurrFeedRate	401
Figure 202: SkwAstroid FC20-Nominal X and Y Feedrate Profiles	402
Figure 203: SkwAstroid FC20 Nominal Tangential Acceleration	402
Figure 204: SkwAstroid FC20 Nominal Rising S-Curve Profile	403
Figure 205: SkwAstroid FC20 Nominal Falling S-Curve Profile	403
Figure 206: SkwAstroid FC10 Colored Feedrate Profile data ngcode	404

Figure 207: SkwAstroid FC20 Colored Feedrate Profile data ngcode	404
Figure 208: SkwAstroid FC30 Colored Feedrate Profile data ngcode	405
Figure 209: SkwAstroid FC40 Colored Feedrate Profile data ngcode	405
Figure 210: SkwAstroid FC10 Tangential Acceleration	406
Figure 211: SkwAstroid FC20 Tangential Acceleration	406
Figure 212: SkwAstroid FC30 Tangential Acceleration	407
Figure 213: SkwAstroid FC40 Tangential Acceleration	407
Figure 214: SkwAstroid FC20 Nominal Separation NAL and NCL	408
Figure 215: SkwAstroid Difference SAL minus SCL for FC10 FC20 FC30 FC40	408
Figure 216: SkwAstroid FC10 FrateCmd CurrFrate X-Frate Y-Frate	409
Figure 217: SkwAstroid FC20 FrateCmd CurrFrate X-Frate Y-Frate	409
Figure 218: SkwAstroid FC30 FrateCmd CurrFrate X-Frate Y-Frate	410
Figure 219: SkwAstroid FC40 FrateCmd CurrFrate X-Frate Y-Frate	410
Figure 220: SkwAstroid FC10 Four Components FeedrateLimit	411
Figure 221: SkwAstroid FC20 Four Components FeedrateLimit	411
Figure 222: SkwAstroid FC30 Four Components FeedrateLimit	412
Figure 223: SkwAstroid FC40 Four Components FeedrateLimit	412
Figure 224: SkwAstroid Histogram Points FC10 FC20 FC30 FC40	413
Figure 225: Plot of Ribbon-10L curve	418
Figure 226: Ribbon-10L Radius of Curvature	418
Figure 227: Ribbon-10L Validation in LinuxCNC	419
Figure 228: Ribbon-10L Direction of Travel 3D	419
Figure 229: Ribbon-10L First and Second Order Taylor's Approximation	420
Figure 230: Ribbon-10L First minus Second Order Taylor's Approximation	420
Figure 231: Ribbon-10L First and Second Order Taylor's Approximation	421
Figure 232: Ribbon-10L Separation SAL and SCL	421
Figure 233: Ribbon-10L Chord-error in close view 2 scales	422
Figure 234: Ribbon-10L Four Components Feedrate Limit	422

Figure 235: Ribbon-10L FrateCommand FrateLimit and Curr-Frate	423
Figure 236: Ribbon-10L FeedRateLimit minus CurrFeedRate	423
Figure 237: Ribbon-10L FC20-Nominal X and Y Feedrate Profiles	424
Figure 238: Ribbon-10L FC20 Nominal Tangential Acceleration	424
Figure 239: Ribbon-10L FC20 Nominal Rising S-Curve Profile	425
Figure 240: Ribbon-10L FC20 Nominal Falling S-Curve Profile	425
Figure 241: Ribbon-10L FC10 Colored Feedrate Profile data ngcode	426
Figure 242: Ribbon-10L FC20 Colored Feedrate Profile data ngcode	426
Figure 243: Ribbon-10L FC30 Colored Feedrate Profile data ngcode	427
Figure 244: Ribbon-10L FC40 Colored Feedrate Profile data ngcode	427
Figure 245: Ribbon-10L FC10 Tangential Acceleration	428
Figure 246: Ribbon-10L FC20 Tangential Acceleration	428
Figure 247: Ribbon-10L FC30 Tangential Acceleration	429
Figure 248: Ribbon-10L FC40 Tangential Acceleration	429
Figure 249: Ribbon-10L FC20 Nominal Separation NAL and NCL	430
Figure 250: Ribbon-10L SAL minus SCL for FC10 FC20 FC30 FC40	430
Figure 251: Ribbon-10L FC10 FrateCmd CurrFrate X-Frate Y-Frate	431
Figure 252: Ribbon-10L FC20 FrateCmd CurrFrate X-Frate Y-Frate	431
Figure 253: Ribbon-10L FC30 FrateCmd CurrFrate X-Frate Y-Frate	432
Figure 254: Ribbon-10L FC40 FrateCmd CurrFrate X-Frate Y-Frate	432
Figure 255: Ribbon-10L FC10 Four Components FeedrateLimit	433
Figure 256: Ribbon-10L FC20 Four Components FeedrateLimit	433
Figure 257: Ribbon-10L FC30 Four Components FeedrateLimit	434
Figure 258: Ribbon-10L FC40 Four Components FeedrateLimit	434
Figure 259: Ribbon-10L Histogram Points FC10 FC20 FC30 FC40	435
Figure 260: Plot of Ribbon-100L curve	440
Figure 261: Ribbon-100L Radius of Curvature	440
Figure 262: Ribbon-100L Validation in LinuxCNC	441
Figure 263: Ribbon-100L Direction of Travel 3D	441

Figure 264: Ribbon-100L First and Second Order Taylor's Approximation	442
Figure 265: Ribbon-100L First minus Second Order Taylor's Approximation	442
Figure 266: Ribbon-100L-First and Second Order Taylor's Approximation	443
Figure 267: Ribbon-100L Separation SAL and SCL	443
Figure 268: Ribbon-100L Chord-error in close view 2 scales	444
Figure 269: Ribbon-100L Four Components Feedrate Limit	444
Figure 270: Ribbon-100L FrateCommand FrateLimit and Curr-Frate	445
Figure 271: Ribbon-100L FeedRateLimit minus CurrFeedRate	445
Figure 272: Ribbon-100L FC20-Nominal X and Y Feedrate Profiles	446
Figure 273: Ribbon-100L FC20 Nominal Tangential Acceleration	446
Figure 274: Ribbon-100L FC20 Nominal Rising S-Curve Profile	447
Figure 275: Ribbon-100L FC20 Nominal Falling S-Curve Profile	447
Figure 276: Ribbon-100L FC10 Colored Feedrate Profile data ngcode	448
Figure 277: Ribbon-100L FC20 Colored Feedrate Profile data ngcode	448
Figure 278: Ribbon-100L FC30 Colored Feedrate Profile data ngcode	449
Figure 279: Ribbon-100L FC40 Colored Feedrate Profile data ngcode	449
Figure 280: Ribbon-100L FC10 Tangential Acceleration	450
Figure 281: Ribbon-100L FC20 Tangential Acceleration	450
Figure 282: Ribbon-100L FC30 Tangential Acceleration	451
Figure 283: Ribbon-100L FC40 Tangential Acceleration	451
Figure 284: Ribbon-100L FC20 Nominal Separation NAL and NCL	452
Figure 285: Ribbon-100L SAL minus SCL for FC10 FC20 FC30 FC40	452
Figure 286: Ribbon-100L FC10 FrateCmd CurrFrate X-Frate Y-Frate	453
Figure 287: Ribbon-100L FC20 FrateCmd CurrFrate X-Frate Y-Frate	453
Figure 288: Ribbon-100L FC30 FrateCmd CurrFrate X-Frate Y-Frate	454
Figure 289: Ribbon-100L FC40 FrateCmd CurrFrate X-Frate Y-Frate	454
Figure 290: Ribbon-100L FC10 Four Components FeedrateLimit	455
Figure 291: Ribbon-100L FC20 Four Components FeedrateLimit	455
Figure 292: Ribbon-100L FC30 Four Components FeedrateLimit	456

Figure 293: Ribbon-100L FC40 Four Components FeedrateLimit	456
Figure 294: Ribbon-100L Histogram Points FC10 FC20 FC30 FC40	457
Figure 295: Plot of AstEpi curve	462
Figure 296: AstEpi Radius of Curvature	462
Figure 297: AstEpi Validation in LinuxCNC	463
Figure 298: AstEpi Direction of Travel 3D	463
Figure 299: AstEpi First and Second Order Taylor's Approximation	464
Figure 300: AstEpi First minus Second Order Taylor's Approximation	464
Figure 301: AstEpi Separation First and Second Order Taylor's Approximation	465
Figure 302: AstEpi Separation SAL and SCL	465
Figure 303: AstEpi Chord-error in close view 2 scales	466
Figure 304: AstEpi Four Components Feedrate Limit	466
Figure 305: AstEpi FrateCommand FrateLimit and Curr-Frate	467
Figure 306: AstEpi FeedRateLimit minus CurrFeedRate	467
Figure 307: AstEpi FC20-Nominal X and Y Feedrate Profiles	468
Figure 308: AstEpi FC20 Nominal Tangential Acceleration	468
Figure 309: AstEpi FC20 Nominal Rising S-Curve Profile	469
Figure 310: AstEpi FC20 Nominal Falling S-Curve Profile	469
Figure 311: AstEpi FC10 Colored Feedrate Profile data ngcode	470
Figure 312: AstEpi FC20 Colored Feedrate Profile data ngcode	470
Figure 313: AstEpi FC30 Colored Feedrate Profile data ngcode	471
Figure 314: AstEpi FC40 Colored Feedrate Profile data ngcode	471
Figure 315: AstEpi FC10 Tangential Acceleration	472
Figure 316: AstEpi FC20 Tangential Acceleration	472
Figure 317: AstEpi FC30 Tangential Acceleration	473
Figure 318: AstEpi FC40 Tangential Acceleration	473
Figure 319: AstEpi FC20 Nominal Separation NAL and NCL	474
Figure 320: AstEpi Difference SAL minus SCL for FC10 FC20 FC30 FC40	474
Figure 321: AstEpi FC10 FrateCmd CurrFrate X-Frate Y-Frate	475

Figure 322: AstEpi FC20 FrateCmd CurrFrate X-Frate Y-Frate	475
Figure 323: AstEpi FC30 FrateCmd CurrFrate X-Frate Y-Frate	476
Figure 324: AstEpi FC40 FrateCmd CurrFrate X-Frate Y-Frate	476
Figure 325: AstEpi FC10 Four Components FeedrateLimit	477
Figure 326: AstEpi FC20 Four Components FeedrateLimit	477
Figure 327: AstEpi FC30 Four Components FeedrateLimit	478
Figure 328: AstEpi FC40 Four Components FeedrateLimit	478
Figure 329: AstEpi Histogram Points FC10 FC20 FC30 FC40	479
Figure 330: Plot of SnaHyp curve	484
Figure 331: SnaHyp Radius of Curvature	484
Figure 332: SnaHyp Validation in LinuxCNC	485
Figure 333: SnaHyp Direction of Travel 3D	485
Figure 334: SnaHyp First and Second Order Taylor's Approximation	486
Figure 335: SnaHyp First minus Second Order Taylor's Approximation	486
Figure 336: SnaHyp Separation First and Second Order Taylor's Approximation	487
Figure 337: SnaHyp Separation SAL and SCL	487
Figure 338: SnaHyp Chord-error in close view 2 scales	488
Figure 339: SnaHyp Four Components Feedrate Limit	488
Figure 340: SnaHyp FrateCommand FrateLimit and Curr-Frate	489
Figure 341: SnaHyp FeedRateLimit minus CurrFeedRate	489
Figure 342: SnaHyp FC20-Nominal X and Y Feedrate Profiles	490
Figure 343: SnaHyp FC20 Nominal Tangential Acceleration	490
Figure 344: SnaHyp FC20 Nominal Rising S-Curve Profile	491
Figure 345: SnaHyp FC20 Nominal Falling S-Curve Profile	491
Figure 346: SnaHyp FC10 Colored Feedrate Profile data ngcode	492
Figure 347: SnaHyp FC20 Colored Feedrate Profile data ngcode	492
Figure 348: SnaHyp FC25 Colored Feedrate Profile data ngcode	493
Figure 349: SnaHyp FC28 Colored Feedrate Profile data ngcode	493
Figure 350: SnaHyp FC10 Tangential Acceleration	494

Figure 351: SnaHyp FC20 Tangential Acceleration	494
Figure 352: SnaHyp FC30 Tangential Acceleration	495
Figure 353: SnaHyp FC40 Tangential Acceleration	495
Figure 354: SnaHyp FC20 Nominal Separation NAL and NCL	496
Figure 355: SnaHyp Difference SAL minus SCL for FC10 FC20 FC30 FC40	496
Figure 356: SnaHyp FC10 FrateCmd CurrFrate X-Frate Y-Frate	497
Figure 357: SnaHyp FC20 FrateCmd CurrFrate X-Frate Y-Frate	497
Figure 358: SnaHyp FC25 FrateCmd CurrFrate X-Frate Y-Frate	498
Figure 359: SnaHyp FC28 FrateCmd CurrFrate X-Frate Y-Frate	498
Figure 360: SnaHyp FC10 Four Components FeedrateLimit	499
Figure 361: SnaHyp FC20 Four Components FeedrateLimit	499
Figure 362: SnaHyp FC25 Four Components FeedrateLimit	500
Figure 363: SnaHyp FC28 Four Components FeedrateLimit	500
Figure 364: SnaHyp Histogram Points FC10 FC20 FC30 FC40	501

LIST OF CODE LISTINGS

3.1	Implementation for Approximate Arc-Length Calculation	84
3.2	Implementation of Exact Chord-length Calculation	85
3.3	Implementation of Approximate Arc-Theta Calculation	86
3.4	Implementation of Approximate Arc-Area Calculation	87
3.5	Snippet of Teardrop Algorithm Summary Output	90
3.6	Snippet of Butterfly Algorithm Summary Output	90
3.7	Snippet of Algorithm Compilation, Binding and Linking	94
3.8	G-code snippet for Arabic calligraphy	104
4.1	Anomaly Run FC30 Teardrop L=0.18 at FC30	205
4.2	SnaHyp Machine Epsilon Problems	213
1	Debian10 HP-Laptop-01 CPU Performance	265
2	Ubuntu20 HP-Laptop-01 CPU Performance	265
3	Debian10 HP-Laptop-02 CPU Performance	265
4	Ubuntu20 HP-Laptop-02 CPU Performance	265
5	Calculation of machine epsilon C code	268
6	Display ranges of floating-point and integer numbers	269
7	Machine epsilon affecting addition and subtraction operations	270
8	Resolving machine epsilon issue	271

LIST OF ABBREVIATIONS

CNC	Computerized Numerical Control
NURBS	Non Uniform Rational B-Splines
EMC	Enhanced Machine Controller
NIST	National Institute of Standards and Technology (USA)
IEEE	Institute of Electrical and Electronics Engineers
NGC	Next Generation Controller
IEEE-1284	Standard for bi-directional parallel port communications between computers
RS274/NGC	CNC control language Recommended Standard 274
OS	Operating System
GPOS	General Purpose Operating System
RTOS	RealTime Operating System
NRTOS	Non-RealTime Operating System
CPU	Central Processing Unit
OMAP	Open Multimedia Applications Platform CPU processor
FPGA	Field Programmable Gate Arrays
RS232	Recommended Standard 232 Serial port communication protocol
PTP	Point To Point
SBC	Single Board Computer
RAM	Random Access Memory
IoT	Internet of Things
VHDL	Verilog Hardware Description Language for FPGA
USB	Universal Serial Bus
UART	Universal Asynchronous Receiver Transmitter
SVG	Scalar Vector Graphics
STL	STereoLithography

LinuxCNC	Linux CNC Control System software
DAQNavi	Data Acquisition Navigator software
Panaterm	Panasonic Terminal servo-motor controller software
FC	Feedrate Command
NFC	Nominal Feedrate Command
LSF	Lamda Safety Factor
V & V	Validation and Verification
SRV	Simulation Run Validation
RRV	Real Run Validation
TIP	Total Interpolated Points
SAL	Sum Arc Length
SCL	Sum Chord Length
SCE	Sum Chord Error
CEE	Chord Error Efficiency
SAA	Sum Arc Area
AEE	Arc Area Efficiency
NAL	Next Arc Length
NCL	Next Chord Length
NAA	Next Arc Area
CSAL	Cumulative Sum Arc Length
CSCL	Cumulative Sum Chord Length
CNAA	Cumulative Next Arc Area
UMP	Universiti Malaysia Pahang
MMU	Multimedia University

Chapter 1

INTRODUCTION

1.1 Introduction

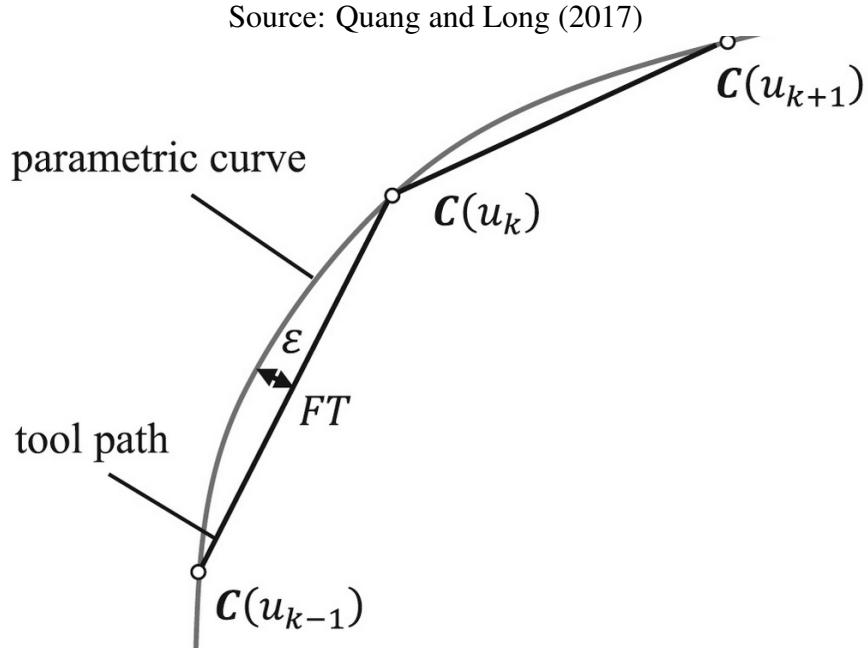
The RS274/NGC standard interpreter is a software system that reads numerical control code in the NGC dialect of the RS274 numerical control language and produces calls to a set of canonical machining functions. The output of the interpreter can be used to drive Computerized Numerical Control (CNC) machines with three to six axes.

There are many merits of parametric interpolation over the traditional linear and circular interpolation, specifically in terms of model representation, feedrate smoothness and application range. One of the major difficulties of parametric interpolation is the feedrate determination that satisfies geometrical constraints and kinematical characteristics of different CNC machine tools.

A majority of CNC systems today supports parametric interpolation because of its various advantages over traditional linear and circular interpolations. The linear interpolator (G01) and the circular interpolator (G02, G03) are the only two interpolators defined in the RS274 standard. References: S.H. Suh and Stroud (2008), and Thomas R. Kramer and Messina (2000). This standard does not have the NURBS interpolator yet that can handle parametric curves and surfaces.

Parametric interpolation is conducted as a point-to-point (PTP) movement in a CNC machine. At the end of each motion it is important that the positional accuracy of the tool relative to the workpiece is achieved, that is, within a specified error tolerance (ϵ). At the end of each motion, the tool performs its required task after which the next motion begins and the cycle repeats until all machining is completed. Reference: Zhong, Luo, Chang, Ding, and Cai (2018).

Figure 1.1: Concept of Chord-error ϵ , Feedrate F, and Interpolation time T



In CNC machine operation, the function of interpolation is to generate coordinated movements to drive the separate axis-of-motions in order to achieve the desired path of the CNC cutting tool relative to the workpiece. Essentially, interpolation generates commands that move motor drives to follow the desired path or trajectory to produce the physical part that is being machined.

Generally, the contour error is a measure of how close the actual tool path is to the desired tool path. For a two-dimensional parametric curve, the point-to-point movement turns the contour error (positional accuracy) to become the chord-error ϵ . This is the error between two interpolated points for the parametric curve. Reference: S. Sun and Xie (2018)

1.2 Problem statement

Generally, for a specific curve path, an increase in feedrate F , will decrease the total number of interpolated points but will increase the chord length (FT), of each arc segment, thereby increasing the chord-error ϵ .

On the other hand, a decrease in feedrate will increase the total interpolated points and reduce the chord-error but will cause the overall machining operation to be slow and time consuming. Thus, there is an interplay between chord-error ϵ , and machine feedrate F . The problem is how to find an optimum balance.

This work proposed a solution by developing a realtime parametric curve interpolation algorithm that constrains both the feedrate and chord-error simultaneously. It was developed and tested for ten(10) different 2-dimensional parametric curves.

The curves were selected based on complex shape characteristics like closed loop curves, open ended curves, symmetric or non-symmetric about the x-axis and y-axis. Some curves have sharp, concave or convex turns including cusps. In addition, the x and y dimensions, that is, their overall sizes vary among the different curves. These are some of the geometrical constraints of the algorithm.

The dynamical and kinematical constraints cover specific CNC machine characteristics like the maximum and minimum allowable machine velocities and accelerations for the different axes of motion, the user specified command feedrate, the jerks and jitter that in combination affects the smooth operation of the machine. References: Yeh (2019), Yu B-F and Yu (2020), and Rob (2022).

1.3 Motivation

Ever since the author was a child, there were always fascinations with regards to moving things and objects, both animated and non-animated. If growth is considered a movement, then humans, animals and plants grow. Whereas the growth of humans and animals are in years, the "growth movements" of vegetation, for example, trees, shrubs and vegetables can be in days, weeks, months and in years, that is, on very different time scales. Humans and animals can move instantly, by choice or instincts.

Inanimate objects, though inherently non moving, can be made to move. Manufacturing robots, vehicles, satellites, spaceships, underwater submarines, drones, missiles and Computerized Numerical Control (CNC) machines, are examples of inanimate objects that can move or move others, with the help of motors, hydraulics, electrical, chemical, wind, wave power, and so on. These inanimate objects can move guided or autonomously by some control system, typically but not necessarily using software.

For example, fighter bombers during the Second World War (early to mid 1940s) were manually controlled using hydraulics by human pilots because there was no software. Thousands of years ago, the ancient Egyptian system for flood and irrigation control of the Nile river uses water wheels and water mills. Software and computing machines were not invented yet so there was no software to use at that time.

The Apollo Control and Guidance System (ACGS developed by MIT) was a crude

combination of basic software, hardware and electronics. The system successfully sent humans to the moon, landed and returned them to earth safely in the late 1960s. This is a very significant landmark in human history.

In particular, CNC machines for example, move electric motors for cutting drills (milling or engraving), lasers and high pressure water jets to cut materials and produce diverse and useful products for humans. Today, the use of software to control movements of some systems is almost ubiquitous.

With the author's having access to a bare-bones prototype CNC machine, the interest grew into the complex world of animating non-animated objects. This led to the current endeavor in CNC machine and its control software, the interpolation program.

Basically, in order to produce the physical part that is being machined, the interpolation program generates commands that move motor drives step-by-step and point-to-point to follow the desired path or trajectory.

Moving inanimate objects is not limited to just cutting things like in the CNC machine. It is also about robots moving around, avoiding obstacles, and performing actions beyond what humans can do, in the air or underwater, in guided or autonomous modes. The possibilities are endless.

1.4 Brief description of this thesis

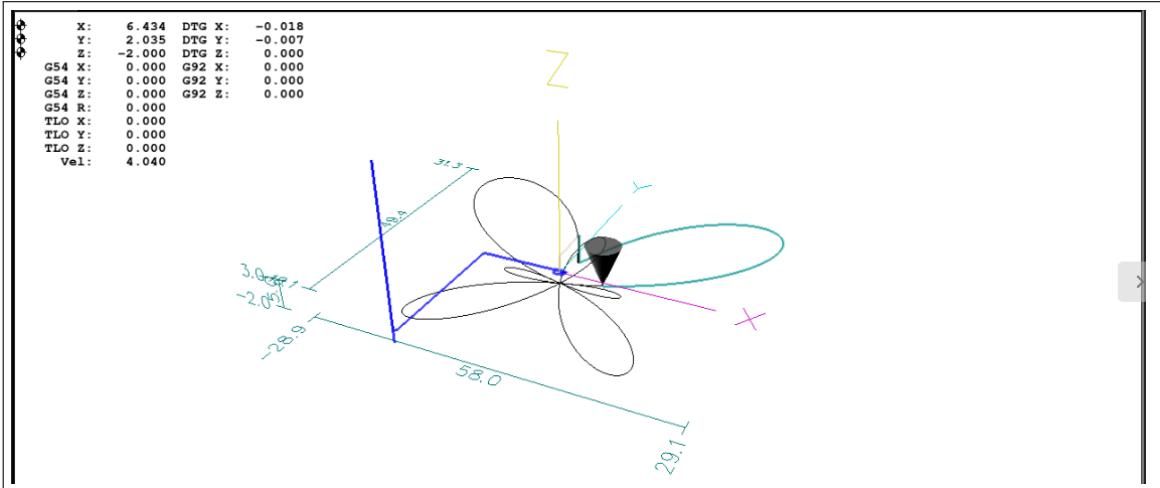
The work in this thesis is about mechatronics and system design. The topic in this work covers five(5) major themes described as follows.

1. Parametric curve - a special mathematical representation of a path trajectory
2. Interpolation - the activity of determining the successive "next-points" in following the specified path trajectory
3. Chord-error - the error between the actual "moved" position and its required position according to the specified path
4. Feedrate - the actual speed of motion along the path.
5. Realtime - the current time in execution by point-to-point moves along the path.

From the above themes, this work is appropriately titled "**Realtime interpolation of parametric curves with chord-error and feedrate constraints.**"

The next section presents an example of the expected results of this thesis.

Figure 1.2: Butterfly Perspective View 3D in LinuxCNC-Axis



1.5 About end results of this thesis

The end results of this thesis can be described as follows. The work in this thesis is about generating a single interpolation algorithm that is successfully applied to ten(10) different parametric curves of various shapes and dimensions. The above figure shows a real live execution of the Butterfly parametric curve (one of the ten curves), by the algorithm developed in this work.

The job of the interpolation algorithm is to generate successive points along the curve trajectory so that the CNC cutting tool (laser cutter) illustrated by the cone in the figure follows the path accurately. The curve begins at parameter $u = 0.00$ (starting point), increasing in steps until $u = 1.00$ (ending point), as the entire Butterfly curve is being followed, that is, from start to finish.

The algorithm uses the second-order Taylor's approximation to calculate the steps ($u\text{-next}$) in parameter u , and at the same time constrains both the chord-error (deviation from the true curve path) to below a set error tolerance (1E-6 mm), and the running feedrate to be very close but below the feedrate limit throughout the full curve path.

The feedrate limit at every parameter u point is calculated by the algorithm based on geometrical, dynamical and kinematical constraints. The constraints comprise 4 different components: user set Feedrate Command FC, minimum and maximum CNC machine axial velocities, minimum and maximum CNC machine axial accelerations, and the geometric factors of the curve path like bends and sharp turns.

The main objective of the algorithm execution is to ensure that the resulting running feedrate (speed motion of the cutting tool) is smooth and continuous, and not exceeding the feedrate limit throughout the full curve path. Note that the feedrate limit varies with u , and thus, the running feedrate also varies, for example, when negotiating curves and sharp bends. The algorithm accomplishes the tool motion strictly without violating both the chord-error and running feedrate constraints.

Since the smoothness of running feedrate is critical to the success of the algorithm, any acceleration jitters (rapid acceleration fluctuations) will result in jerky machine feedrates. The algorithm execution was able to completely avoid this situation.

1.6 Scope of work for this thesis

The scope of work for this thesis can be described as follows:

1. Selection of ten(10) different parametric curves and their equations, each curve with varying features, shapes and dimensions.
2. Setting up the feedrate (velocity) constraining dynamic equations for allowable CNC machine parameters like the maximum and minimum axial velocities, and maximum and minimum axial accelerations.
3. Setting up the chord-error constraining equations involving geometric and kinematic properties of the ten(10) different parametric curves.
4. Development of a realtime algorithm that generates interpolated points which traverses each curve, such that the chord-error and feedrate at all points are simultaneously constrained.
5. Execution of reports that show the interpolation algorithm performs correctly as designed, generating interpolated points and feedrates that are smooth and continuous for all ten(10) parametric curves selected.
6. In addition, the reports on algorithm implementation show that the absolute constraints on chord-error and feedrate are not violated throughout the full trajectory of each curve.
7. The execution of the algorithm on all ten(10) curves also show that acceleration jitters do not occur, and the acceleration is always maintained between the designed maximum and minimum limit.
8. The algorithm also generates RS274/NGC G-Code file for validation and verification on the CNC machine running the LinuxCNC-Axis control application.

9. For the performance assessment of the developed algorithm, four different metrics were created. The four metrics are:
 - SCE/TIP, that is the ratio of total sum-chord-error divided by the total number of interpolated points.
 - SCE/SCL, that is the ratio of total sum-chord-error divided by the total sum-chord-length.
 - SAA/SCL, that is the ratio of the sum-arc-areas divided by the total sum-chord-length.
 - $100(\text{SAL}-\text{SCL})/\text{SAL}$, that is the ratio of the difference between the sum-arc-length and the sum-chord-length, divided by the sum-arc length and multiplied by 100 to represent it in percentage form.
10. The results in the main document of this thesis discuss the full outputs of the Teardrop curve resulting from the algorithm execution. The outputs for the rest of the nine(9) selected curves are provided in their respective appendices.

1.7 Organization of this thesis document

Chapter 1 Introduction, covers the basic ideas on parametric interpolation, Computerized Numerical Control (CNC) machines, the G-code standards for these CNC machines, chord-error and machine feedrates associated with CNC machining operations. The introduction also includes the problem statement and the motivations that led to this work. Finally, the organization structure of this thesis is documented.

Chapter 2 Literature Review, involves the topics on mathematical parametric representation of curves and surfaces in NURBS, the general ideas on C^N continuity of curves and surfaces and, the advantages of parametric representations. This chapter also includes the concepts of interpolation of parameteric curves. It is followed by a literature review of published sources for previous works involving interpolation of parametric curves and surfaces. At the end of this chapter, the ideas of NURBS and its relationship to CNC G-code programming languages were discussed.

Chapter 3 Methodology, begins with describing the processing steps involved in parametric curve interpolation, and include the characteristics and presentation of mathematical equations for the ten(10) parametric curves selected for this work. It is followed by the discussion on chord-error (*epsilon*) concepts, the chord-error minimization, feedrate maximization, and the proposed combined chord-error and feedrate constraints to be enforced simultaneously. Next, the criteria of convergence is stated in the brief algorithm design for this work.

Chapter 3 continues with the mathematical derivations of equations to be computed in the algorithm. This includes equations for the radius of curvature (*rho*), chord-error (*epsilon*), current feedrate (*curr_frate*). The derivations of the first and second order iterative Taylor's expansion followed. This chapter then continues with the calculations for the next interpolation point. The discussions and equations for the various feedrate limits (*frate_limit*), consisting of the combined dynamical and geometrical constraints were presented. The main program for the interpolation was described through a flowchart. The equations and flowcharts for the rising and falling of feedrates using the sigmoid S-curves were derived, discussed and presented. Chapter 3 ends with the presentation of a flowchart for the balance of chord-error and feedrate combined constraints.

Chapter 4 Results, presents the final outcomes of the entire work for this thesis. Out of the ten(10) parametric curves studied in this work, the full results of only one(1) of those curves, for illustration, are presented in the main document. The full results for the rest rest of the curves (about 20 pages per curve) are provided in the respective append-

dices. The full results for the rest of the curves can be compiled into a separate, second document, considered an annex to the main document for this thesis.

The main results presented in Chapter 4, for the illustrative curve, comprise profiles for feedrate, chord-error, interpolated step size, radius of curvature, feedrate limit constraints, and tangential accelerations. For overall algorithm performance analyses, a comprehensive table collection of execution data was compiled for all of the ten(10) curves. The data comprise important statistics like total interpolated points, the histogram of interpolated points against the parameter range, the histogram of the chord-errors against the parameter range, the feedrate points above and below feedrate limits, the sum total of chord errors (total accumulated error), the sum of chord lengths (total path length traversed for each curve), and so on.

One interesting performance result for each curve is the ratio of total accumulated error against the total path length traversed. This is a meaningful performance measure for the interpolation algorithm since it measures the amount of chord-error generated by the algorithm per unit length of curve traveled.

The total interpolated points is not a meaningful measure because each curve has a different total length. The results also provides trending data for algorithm executions against four(4) different feedrate command (FC10, FC20, FC30 and FC40) values for each of the curve. The interpretation of the total number of interpolated points for each FC value for comparison becomes meaningful.

Chapter 5 contains the conclusions of this research, sharing of some lessons learned and recommendations for future work.

Chapter 2

LITERATURE REVIEW

2.1 Parametric Representation of curves and surfaces

The standard for describing and modeling curves and surfaces in computer aided design (CAD) and computer graphics is NURBS, or NonUniform Rational B-Splines. Extensive mathematical coverage describing NURBS are found in the seminal books of Rogers (2001) and, Piegl and Tiller (1997).

Essentially, NURBS describe parametric curves and surfaces. Curves and surfaces are mathematically represented either explicitly, implicitly or parametrically.

In general, a parametric curve representation of a 3D curve takes the mathematical functional form of $C(x(u), y(u), z(u))$ where $x(u)$, $y(u)$, and $z(u)$ are mathematical functions in the independent parameter u , bounded by a range $u_{min} \leq u \leq u_{max}$.

By extension, a parametric surface representation of a 3D surface takes the form of $S(x(u, w), y(u, w), z(u, w))$ where $x(u, w)$, $y(u, w)$, and $z(u, w)$ are mathematical functions in two independent parameters u and w . The parameters u and w must also be bounded.

When compared to either explicit or implicit formulations, this parametric representation is extremely flexible. The representation is axis independent, easily represented by multiple-valued functions and, can have infinite derivatives. Furthermore, to have more degrees of freedom independent parameters can be added.

In practice, curves and surfaces are generally bounded. When either an explicit or implicit representation is used, imposing the boundaries is awkward. In contrast, the boundaries for a parametrically represented curve or surface are provided by the restrictive

parameter ranges. In addition, the parameter range for a parametric curve also specifies a natural traversal direction along the curve.

As an example, for a 3-dimensional curve with independent parameter u in the range $u_{min} \leq u \leq u_{max}$, the curve direction will be from the point $C(x(u_{min}), y(u_{min}), h(u_{min}))$ to $C(x(u_{max}), y(u_{max}), h(u_{max}))$ as u increases from u_{min} to u_{max} .

Note that specifying a curve requires one parameter while a surface requires two parameters. Also a parametric curve with the function $h(u)$ being zero for all values of parameter u means the curve is 2-dimensional or a curve in the x-y plane.

2.2 Continuity of curves and surfaces

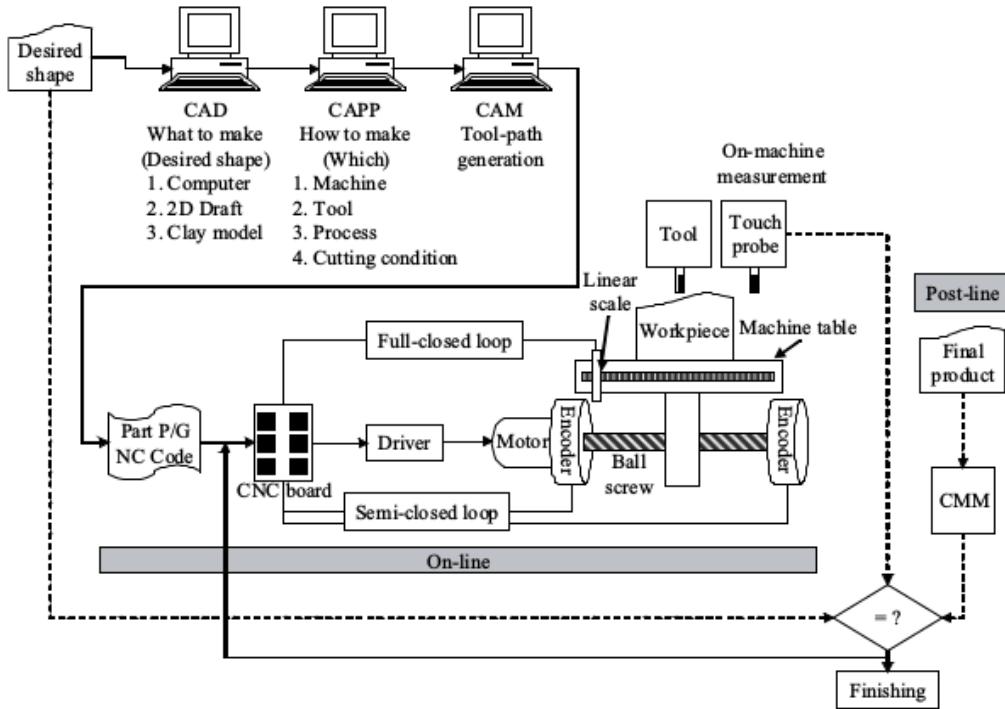
There are two kinds of continuity, or smoothness, associated with parametric curves and surfaces known as geometric continuity G^n , and parametric continuity C^n , where n is the order of the derivative. Simplistically, you can think of geometric continuity as physical and parametric continuity as mathematical. Geometric continuity is less restrictive than parametric continuity. This means that if it is C^n continuous, then it implies G^n continuous, but not the opposite. The fundamental difference between geometric G^n and parametric continuity C^n can be illustrated as follows.

Given a parametric vector-valued function, $P(u)$, over some parameter range u describing a curve, then any given value of u represents a specific point on the curve. The first derivative, $P'(u)$, represents the velocity of a point as it moves along the curve, and the second derivative $P''(u)$ represents the acceleration.

If the curve is C^1 continuous at a join, then both the direction and magnitude of the tangent vector are equal across the join; and the point smoothly transitions from one curve segment to the next. If, however, the curve is only G^1 continuous at the join, then as the point transitions across the join the direction of motion does not change (tangent vector direction), but there is an instantaneous change in speed (tangent vector magnitude) that represents an instantaneous acceleration as the point transitions across the join. If the curve represents a path trajectory for a CNC machine, the abrupt velocity (magnitude) change will cause a sudden jerk.

Note that C^0 means the parametric continuity of the curve itself, and C^1 and C^2 , the continuity of its first and second derivatives, respectively. All of the parametric curves selected and used in this work are C^2 continuous.

Figure 2.1: Basic architecture for a typical CNC system



2.3 Computerized Numerical Control (CNC) Systems

A typical machine architecture for a 3D-four-axis machine is shown in Fig [2.1]. The 3D axes corresponds to the x, y, and z motion axes while the fourth is the rotating axis of the spindle cutting tool. Fig [2.2] shows the mechanism of the axial motions and the rotational motion of the spindle while Fig [2.3] shows the typical software control flow diagram for a typical CNC machine.

CNC manufacturing operations involve the generation of reference signals describing the geometrical parts to be machined and the control of the machine such that it follows those reference signals. In modern CNC machines, the generation of reference signals, construction and execution of control loops are accomplished in software within a computer. Fig [2.4] shows the block diagram for the functional architecture of a typical CNC system.

A modern commercial CNC 3D-four(4)-axis milling machine is shown in Fig [2.5]. The fourth axis is the rotational axis for the spindle cutting tool. An example of a a commercial CNC lathe machine is shown in Fig [2.6]. Modern commercial CNC machines today have extended milling and turning to 9-axes, for example, the DN Solutions PUMA SMX3100ST combined milling and turning machine. TitanCNC (2021) shows a YouTube video of this 9-axes CNC in action.

Figure 2.2: Typical Servo drive and spindle driving mechanism

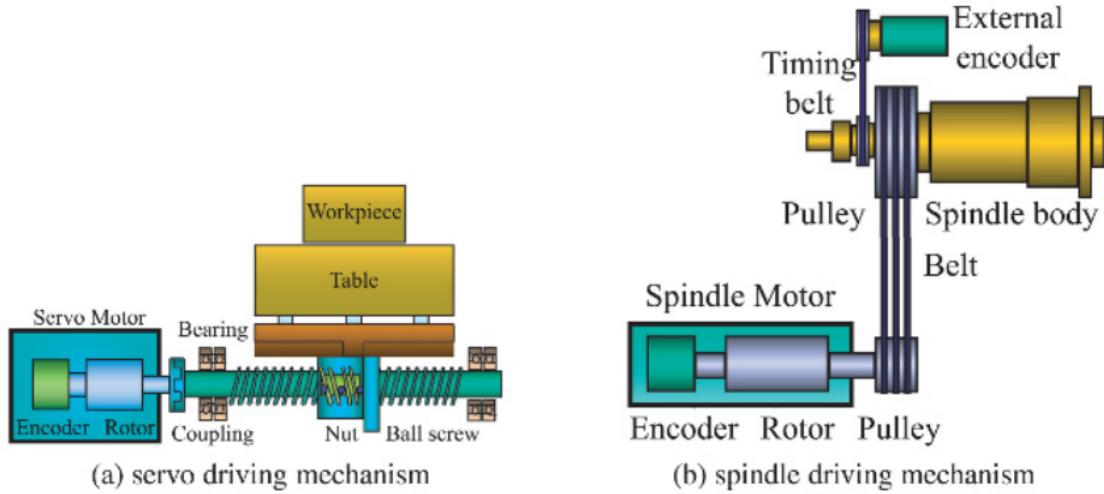


Figure 2.3: Typical CNC Software control flow diagram

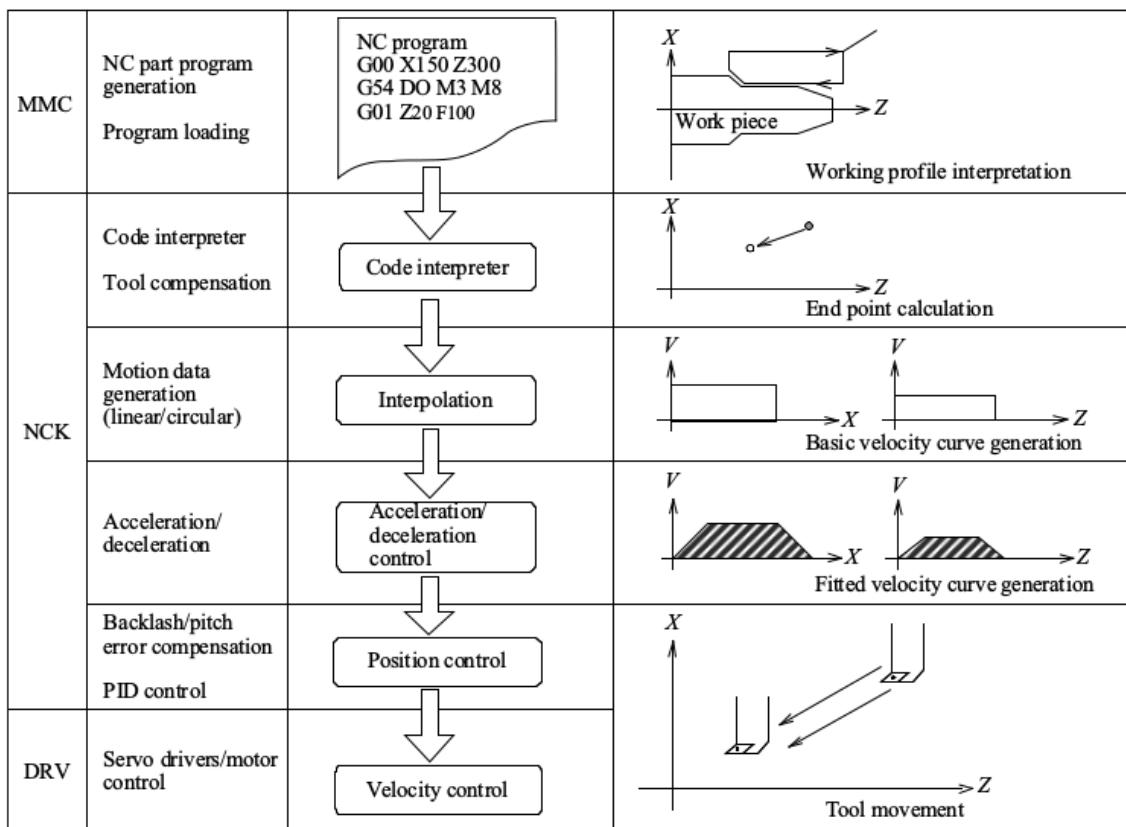


Figure 2.4: Typical CNC Functional Architecture

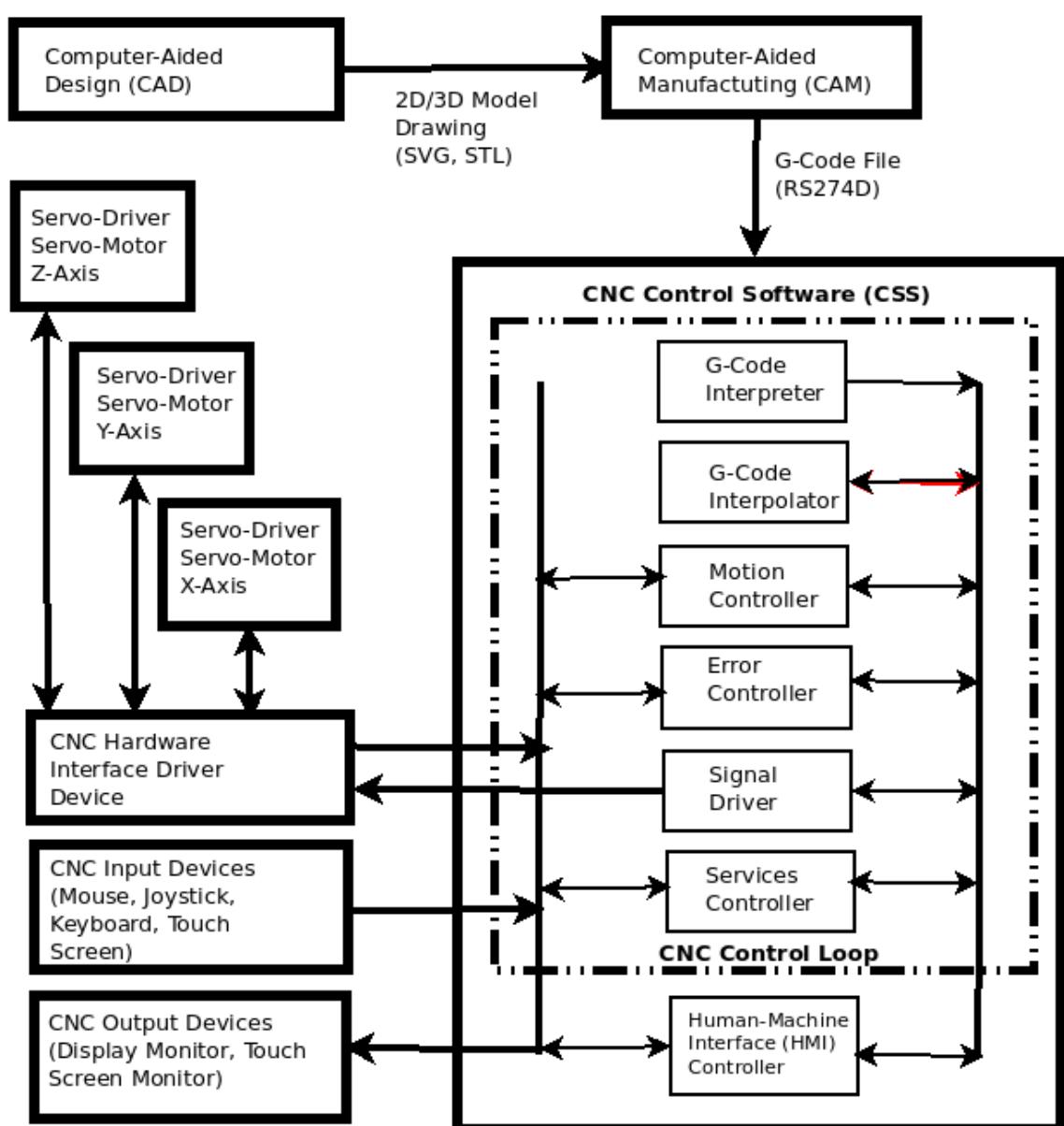


Figure 2.5: Typical CNC Commercial Milling machine





Figure 2.6: Typical CNC Commercial Lathe machine

2.4 Advantages of Parametric Representations in CNC

Parametric interpolation has many merits over the traditional linear (G01) and circular interpolation (G02 and G03) in terms of model representation, feedrate smoothness and application range. Jin, He, Fu, Lin, and Gan (2014) reported some advantages below.

1. In parametric interpolation, the geometrical information of the machining paths can be totally as well as accurately transferred to the CNC systems without any approximation errors and data loss. This error and loss may occur in conventional linear and circular interpolation.
2. The parametric interpolator code only needs some critical parameters of the machining contours (i.e. control points, knot vectors, weights as in NURBS). Such a transmission mechanism can guarantee the efficiency of interaction between the host and slave machines.
3. In parametric interpolation, feedrate continuity and smoothness are achieved effectively since the joints between tiny segments do not require repeated acceleration-deceleration processes. In traditional interpolation methods repeated acceleration-deceleration may not be avoidable.
4. Parametric interpolation is still possible in conventional CNC systems after some developments of the machining parts. This may include approximating tiny parts into curve segments and optimization with parametric curves between them.

2.5 Interpolation of parametric curves

The problem statement for interpolation of a parametric curve is:

Given the parametric curve $C(u)$, determine $u(k)$ for the k_{th} interpolation period, the re-parametrization of u with time t is required, that is, solve the function $u = u(t)$, where u is the independent parameter, and t is time.

In other words, it means at a point on the parametric curve $C(u)$, find the next u point. The next u point is a function of time t .

Since the machine moves with time, the re-parametrization from t to u should be subjected to the machine dynamic constraints and contour error tolerance. It can be described as follows.

1. The machine axial velocities and accelerations should be limited to avoid saturating the axes. These are the first and second derivatives of the corresponding parametric function $C'(u)$ and $C''(u)$ with $u = u(t)$, over time, respectively.
2. The limitations include satisfying specific machine characteristics like startup torque, maximum and minimum velocity (feedrate), maximum and minimum acceleration (jerk), and encoder resolution (incremental motion stepping).
3. In order to guarantee a smooth trajectory motion profile the jerk should be limited or completely eliminated. This ensures a smooth feedrate profile.
4. The contour error ϵ increases with increasing feedrate, so the feedrate should be limited to achieve high contour accuracy. There should be an optimum strategy to balance contour error and machine feedrate.

The function of the real-time interpolator in a computer numerical control (CNC) machine is to compute a reference point in each sampling time interval T (for example, 0.001 s), of the servo system, based on a prescribed tool path (curve) and feedrate data.

By comparing the actual instantaneous machine position (measured by encoders on the machine axes) with the reference point, accurate closed-loop control of the machine position and speed may be obtained.

2.6 Previous works on parametric interpolation

Probably, Koren (1976) was the first to introduce the general concept of interpolation for a CNC system. Later, Koren, Lo, and Shpitalni (1993) and Shpitalni, Koren, and Lo (1994) extended the idea to basic approximation principles for parametric curves.

The two common technical routes that have been developed for the interpolation of parametric curves are the arc length parametrization and the recursive/iterative Taylor's expansion.

It was reported by R. Farouki and Tsai (2001) that near arc length parametrization is possible with some numerical methods. However, it is computationally intensive with the approximation error accumulating along the curve. This especially occurs in curves with large curvature variations like sharp turns, and with uneven parameter computations.

In order to realize parametric interpolation, researchers developed a wide variety of methods to achieve better machining qualities. The common adopted approaches for interpolating parametric curves were based on Taylor's expansion, for example, Cheng, Tsai, and Kuo (2002).

R. T. Farouki (2021) considered realtime interpolators based on Taylor series to possess two inherent shortcomings. Firstly, the occurrence of truncation errors because only a finite (small) number of terms can be employed, and secondly, for variable feedrates, the coefficients of higher-order terms, defined by successive chain-rule differentiation, become increasingly complex and cumbersome, especially when involving computationally intensive expressions.

A parametric interpolation method based on prediction and iterative compensation was proposed by Ni, Zhang, Chen, Hu, and Liu (2019) where the feedrate fluctuation and Taylor's expansion were analyzed to reduce the calculation accuracy of truncation errors caused by neglecting the high-order terms.

Another method for parametric interpolation is named 'guide curve based'. It was developed by Sun, Wang, and Guo (2006). This method implements the near arc length parameterization. Essentially, the detection of corners and key regions are done offline, and this approach schedules feedrate based on a guide curve without a look-ahead facility.

Although available CNC interpolators for parametric curves generally achieve contouring position accuracy, the specified feedrate, which dominates the quality of the machining process, is not guaranteed to be smooth during motion. Severe speed fluctuations may incur tool chatter or breakage, especially in high speed machining. Since feedrate control is crucial, Yeh and Hsu (1999) for example, developed a speed-controlled interpolator for machining parametric curves.

Later, Yeh and Hsu (2002) considered interpolation of parametric curves by adapting the feedrate with a confined chord error while Nam and Yang (2004) proposed an approach for parametric interpolation that allows the feedrate profile to be dynamically adjusted according to geometrical path constraints.

Tracking error and contour error are two very important aspects that need to be effectively handled by any CNC servo control system. Ramesh, Mannan, and Poo (2005) reviewed the various techniques that have been developed in minimizing and eliminating tracking and contour errors.

A much later review, 13 years after, was conducted by Jia, Ma, Song, Wang, and Liu (2018) on the subject of contouring-error reduction method in multi-axis CNC machining. The review covers both three-axis and five-axis contouring-error reduction methods. The methods comprise various offline contouring-error reduction, interpolator designs for contouring-error reduction, cross-coupling schemes for direct contour control and contour control algorithms. As an example, M. Chen and Sun (2019) considered contour error-bounded parametric interpolator with minimum feedrate fluctuation.

The paper by Lee et al. (2018) reviews recent progress of control technologies for precision machining using CNC in the area of interpolation, contour control and compensation. In terms of interpolation several corner blending methods and parametric curves are introduced and the characteristics of each method discussed. In addition, contour control algorithms recently developed for multi-axis machines are also reviewed.

In another direction, Bhattacharjee, Azeem, Ali, and Paul (2012), instead of using Taylor's expansion approximation, employed the classical fourth-order Runge-Kutta (RK) method using only the first derivative to generate successive parameter values for the calculation of x, y, z coordinates of interpolated points. It was done in order to avoid calculating higher derivatives and make the interpolation calculations simpler.

On parallel implementation, Fu, Li, and Fu (2016) developed a parallel CNC system architecture based on Symmetric Multi-Processor (SMP). This subject is of interest to this author. The author's involvement on parallelism in CNC is discussed in the next section, where FPGA (Field Programmable Gate Arrays) programming was used to drive the CNC machine.

2.7 Related CNC work by the author

Low end research CNC machines, like the CNC UMP Model 2008 in Fig [2.7], can be controlled and driven using small devices and (IoT) gadgets, like desktops, laptops, computer extension boards and single-board-computers (SBC).

Kalimbetov and Yusoff (2012) used C/C++ and Python codes on a Linux box driving the UMP CNC model 2008 machine in realtime using a customized parallel port software driver. Comparisons were conducted running the CNC on both realtime (RTOS) and non-realtime (NRTOS) operating systems.

Ariffin and Yusoff (2014) used the USB based Arduino Due extension board to drive the UMP CNC machine. This resource limited hardware (only 512 KB onboard user Flash memory) required a special technique in writing the C/C++ codes for the CNC software driver.

Abdelgader and Yusoff (2014) used a combination of gadgets to drive the CNC machine, using the computer laptop, the proprietary Heber X10i extension board and the Raspberry Pi 2 single board computer (SBC). The Raspberry Pi a low cost, credit-card sized, full board computer was successfully used to drive the CNC machine. Since, the memory addresses of the driving pins in the Heber X10i hardware are separated, a special technique was developed to get the memory register addresses contiguous.

Ahmad and Yusoff (2017) later used the Raspberry Pi 3 Model B, an updated version to drive a CNC machine in Real Time. An output for this work is shown in Fig [2.9]. The step by step CNC process from start to finish (capture image, convert image to G-code, drive G-code on UMP CNC machine) executed by Ahmad and Yusoff (2017) is shown in an interesting video accessible at the link: [CNC Rpi3 Real Time, 5m:46s]. The video is demonstrated with a beautiful background rendition of the song "Jalur Gemilang", a snapshot of it is provided in [2.8].

Selvarajah and Yusoff (2015) deployed the Beagle Board xM single board computer (SBC), to drive the CNC machine. This Beagle Board xM is a variation of SBC gadgets with more capabilities than the Raspberry PI. This board used the OMAP (Open Multimedia Applications Platform) processor, which is capable of video and image processing. OMAP is mostly used in commercial cell phones.

The highlight is the work of Teh and Yusoff (2016) that engaged the Nexys-3 Spartan-6 FPGA board to develop a closed-loop feedback system that controlled the CNC machine. An image of this quite expensive board is provided in Fig [2.10] at the bottom right. the green board with a USB-UART connection.

FPGAs are truly parallel in nature, so different processing operations do not have to compete for the same resources. In addition, the inherent parallel execution of FPGAs allows for independent pieces of hardware logic to be driven by different clocks. Depending on design, FPGA processors execute multiple tasks at any one time. This is unlike typical Intel x86 or ARM processors that execute line by line.

FPGAs are powerful pieces of technology that allow users to implement custom hardware (not software) designs. There are several FPGA programming languages available, but the four most popular are VHDL, Verilog, SystemVerilog, and C/C++.

This project utilized a mixed of VHDL and C/C++ languages to program the Nexys-3 hardware that generated the CNC driver codes on the the board. A G-code file on the computer was first converted to a signal file, reference Fig [2.11]. Then, the file is transmitted from the computer to the Nexys-3 board via USB-UART serial protocol. The received signals are then sent by the Nexys hardware to the three(3) x, y, z axial motors of the CNC machine, with the three tasks running parallel in time. Since parallel execution is natural to FPGA operation, the project goal of true parallelism was achieved.

Real live Youtube videos of successful CNC executions can be accessed at the following links: [CNC FPGA-Video1, 0:56s], [CNC FPGA-Video2, 0:58s] and [CNC FPGA-Video3, 4m:00s].

The acquired detailed knowledge, skills and understanding of these diverse raw hardware and their environments, as mentioned in the projects above, is critical to the success of the present author's thesis.

Figure 2.7: CNC UMP Model 2008 - University Malaysia Pahang

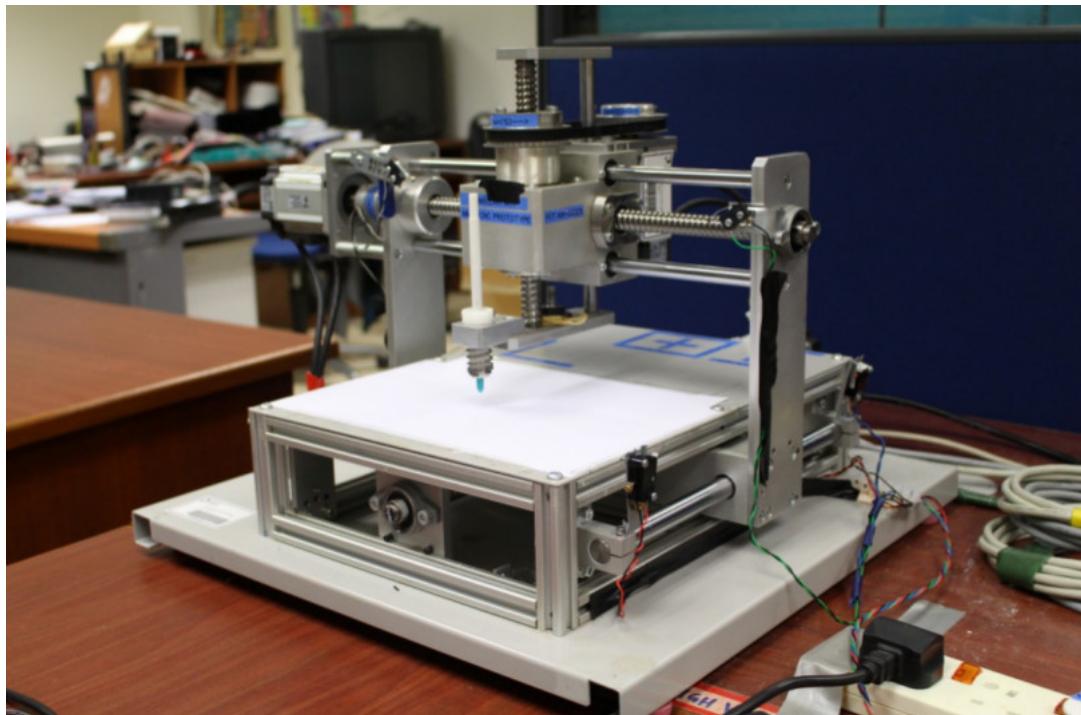
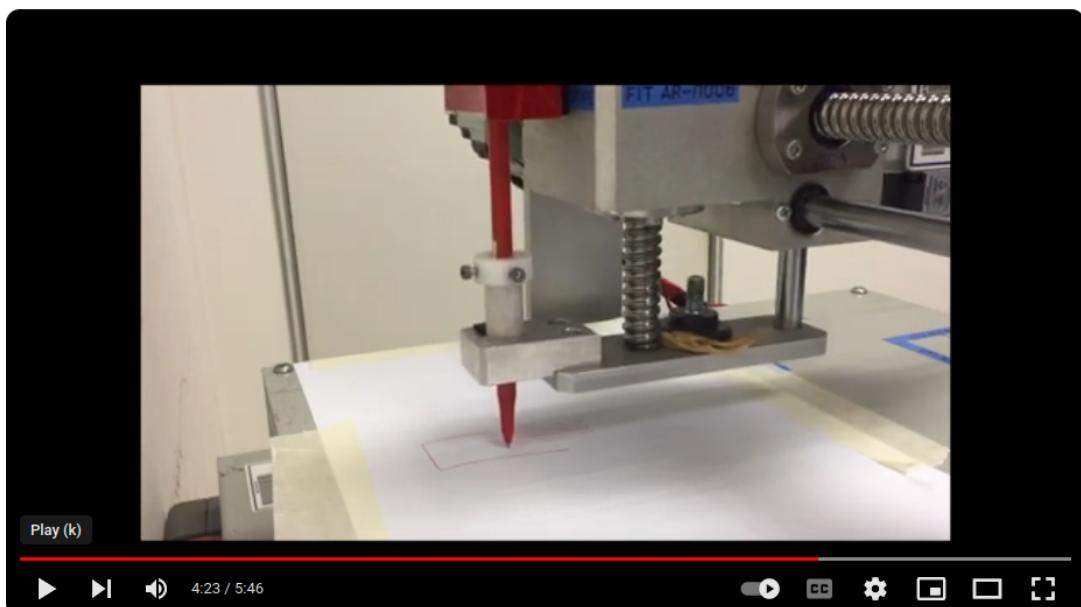


Figure 2.8: Youtube: CNC Jalur Gemilang Rendition Video by Ahmad (2017)



Raspberry Pi Project CNC Machine 2

A Asyru1 Ahmad
43 subscribers

Subscribe

DOWNLOAD AS: ▾

1

Share

Download

...

Figure 2.9: Proton logo output of UMP CNC machine by Ahmad:2017

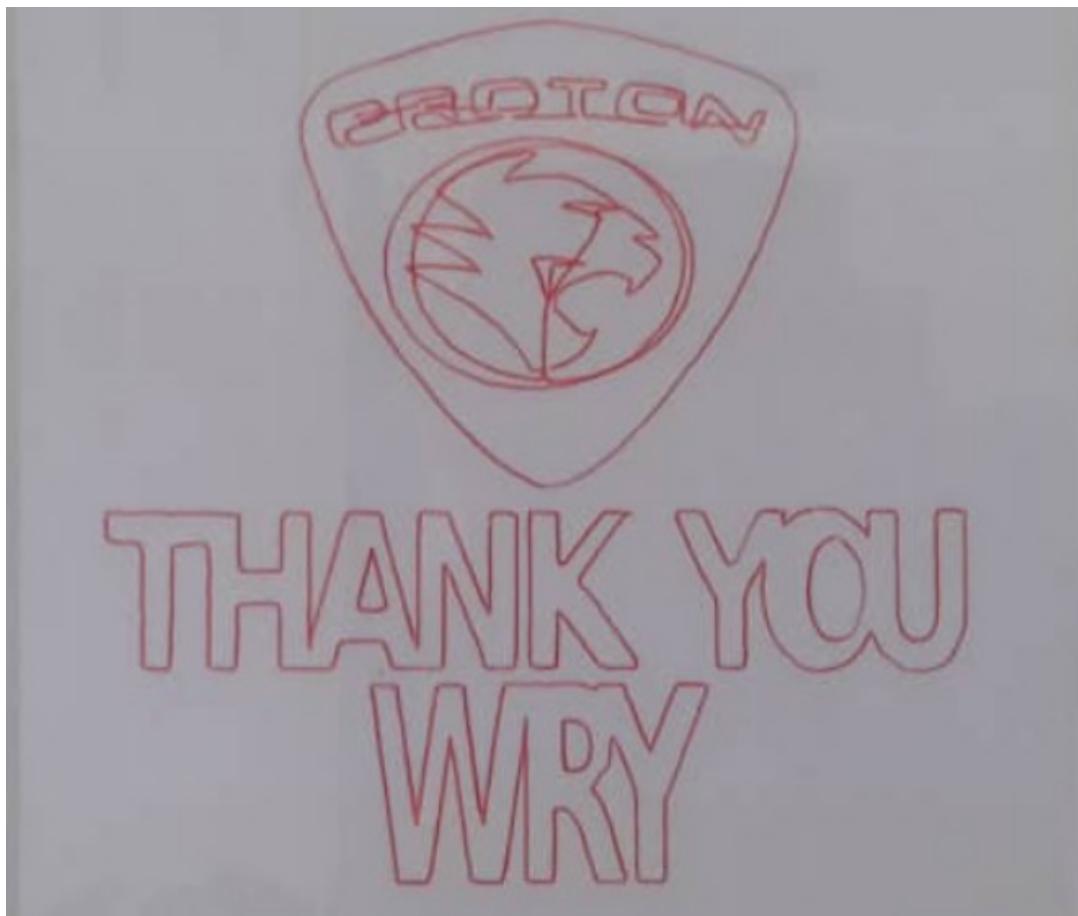
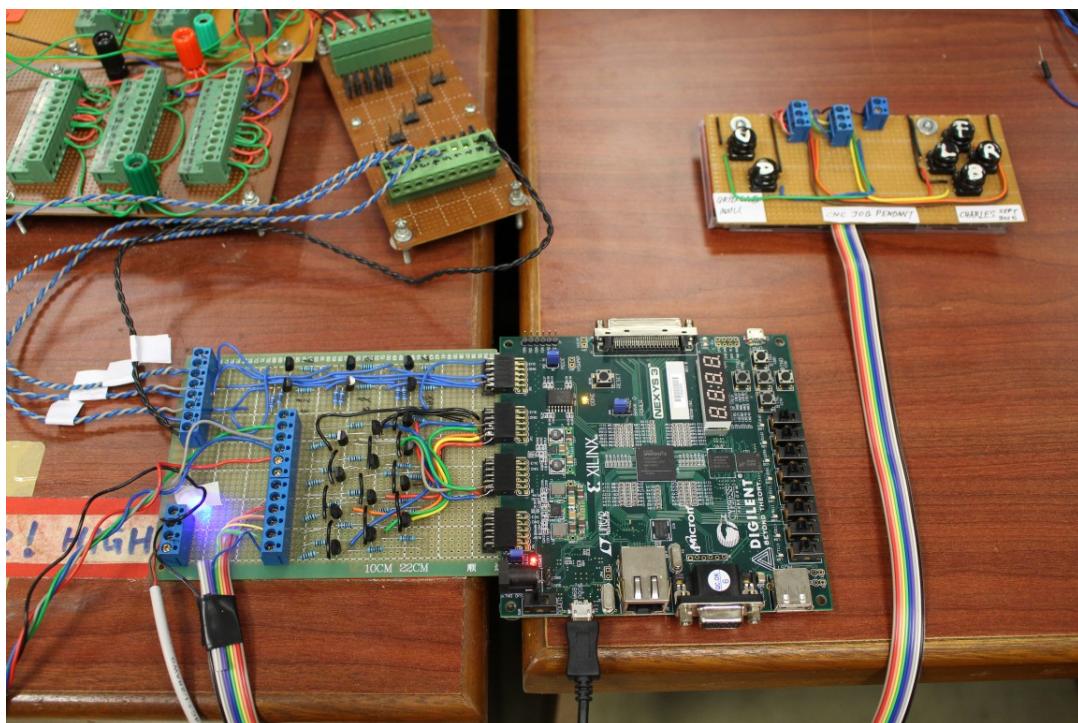


Figure 2.10: Setup of Nexsys-3 Spartan-6 Xilinx board by Teh:2016



2.8 Comparisons previous works with this thesis

Previous works had definitely looked into the issues of chord-error and feedrate in parametric curve interpolations, but in a different perspective compared to the work in this thesis.

The early paper by Yeh and Hsu (1999) formulated real-time CNC interpolators for variable feedrates along parametric curves. Later, R. Farouki and Tsai (2001) pointed an erroneous coefficient for the quadratic term in the paper.

It was mentioned by R. Farouki and Tsai (2001) that, Pythagorean-hodograph (PH) curves admit closed-form analytic reductions of the interpolation integral, yielding real-time interpolators that are essentially exact, and remarkably versatile in terms of the repertoire of feedrate variations (with time, arc length, or curvature) they accommodate. The work in this thesis, also involves feedrate variations with time, arc length, and curvature however, it does not involve Pythagorean-hodograph (PH) curves.

R. T. Farouki (2021) considered the successive applications of the differentiation chain rule which are necessary to determine Taylor series coefficients beyond the linear term as being cumbersome, so the use of Richardson extrapolation as a simple means to compute rapidly convergent approximations to the higher-order coefficients was introduced. The work in this thesis, however, does not involve Richardson extrapolation.

The work by Jin et al. (2014) breaks parametric interpolation into rough interpolation and fine interpolation. According to Jin et al. (2014), rough interpolation is about feedrate adjustments between two successive interpolation periods, while fine interpolation is about adjusting the feedrate to comply with the geometrical constraints and kinematical characteristics of the CNC machine. The work in this thesis however, implements feedrate adjustments immediately in one interpolation period taking into account of both geometrical and kinematical constraints simultaneously.

The work by Zhong et al. (2018) does not impose an upper or lower limit for the chord-error. The values of chord-error are considered as whatever resulting from the algorithm running at the particular feedrate. In addition, Zhong et al. (2018) also allows for feedrate speedup to the value of command feedrate, for example, when it foresees a near straight line coming ahead, and this breaks the absolute feedrate limit constraint.

As a comparison, the work in this thesis imposes an upper limit for the chord-error, and strictly preserves both the chord-error and feedrate limit constraints at every point on the curve path, simultaneously. In addition, for feedrate smoothness throughout the entire traversal of the curve path, this thesis imposes a strict condition that does not allow acceleration jitters. This was not a condition mentioned in previous cited works.

2.9 NURBS and G-Code programming

NURBS are a type of spline, which is a curve or surface defined by a set of control points and a basis function. Unlike other splines, NURBS can have rational weights associated with each control point, which allow them to represent conic sections and other shapes that are not possible with polynomial splines. NURBS also have the advantage of being invariant under affine transformations, such as scaling, rotation, and translation, which makes them easier to manipulate and transform. NURBS can also be used to approximate any shape with arbitrary precision, by adjusting the number and position of the control points and the degree of the basis function. Reference: A. A. CollabCNC (2023)

One of the main challenges of using NURBS in CNC programming is the compatibility and interoperability between different software and hardware systems. Not all CAD/CAM software can export or import NURBS data, and not all CNC machines can process or interpret NURBS data. Therefore, you may need to use different formats, standards, or protocols to exchange NURBS data between different platforms, such as IGES, STEP, DXF, or G-code. Another challenge is the quality control and verification of the NURBS data and the machined parts. You may need to use special tools or methods to check the accuracy, smoothness, and consistency of the NURBS data and the output, such as error analysis, simulation, or inspection. B. A. CollabCNC (2023) discusses how to evaluate the accuracy and quality of NURBS machining results.

Low end CAD/CAM systems do not generate NURBS G-Code. However, NURBS have been used by high end systems CAD systems for some time. That is the main reason why it is natural that CNCs should be able to employ tool paths that are also defined in terms of NURBS.

This work covers only RS274D NGC G-Code files because the format is the base standard for G-Codes supported by all machines used in the CNC industry. This work does not include Scalar Vector Graphics (SVG) or STereoLithography (STL) files generated by CAD applications. It also excludes the generation of G-Codes from CAM processing of SVG or STL input files.

Using NURBS interpolation requires a CNC machine capable of handling NURBS G-Code generated tool paths. Since the NURBS G-Code format is proprietary and only used in the high end FANUC CNC machines. It should be noted that NURBS G-Code files are not the same as NURBS interpolation method. A. A. CollabCNC (2023), for example, discusses on how to integrate NURBS with other CNC programming methods and techniques.

Very recently, as of July 17, 2023, Hu et al. (2023) published a method for calculating interpolation points of NURBS curves based on chord length-parameter ratio.

The construction and manipulation of NURBS geometries is based on a structure with the following fields:

- (1) number: the number of control points.
- (2) coefs: control points coordinates (for NURBS also the weights).
- (3) order: the degree plus one.
- (4) knots: the knot vector in each direction

2.10 Programming Languages for NURBS

Traditionally, NURBS algorithms were developed using compiled C and C++ programming languages which are tedious and complicated to setup and use. It is available at the link [C-Codes for NURBS].

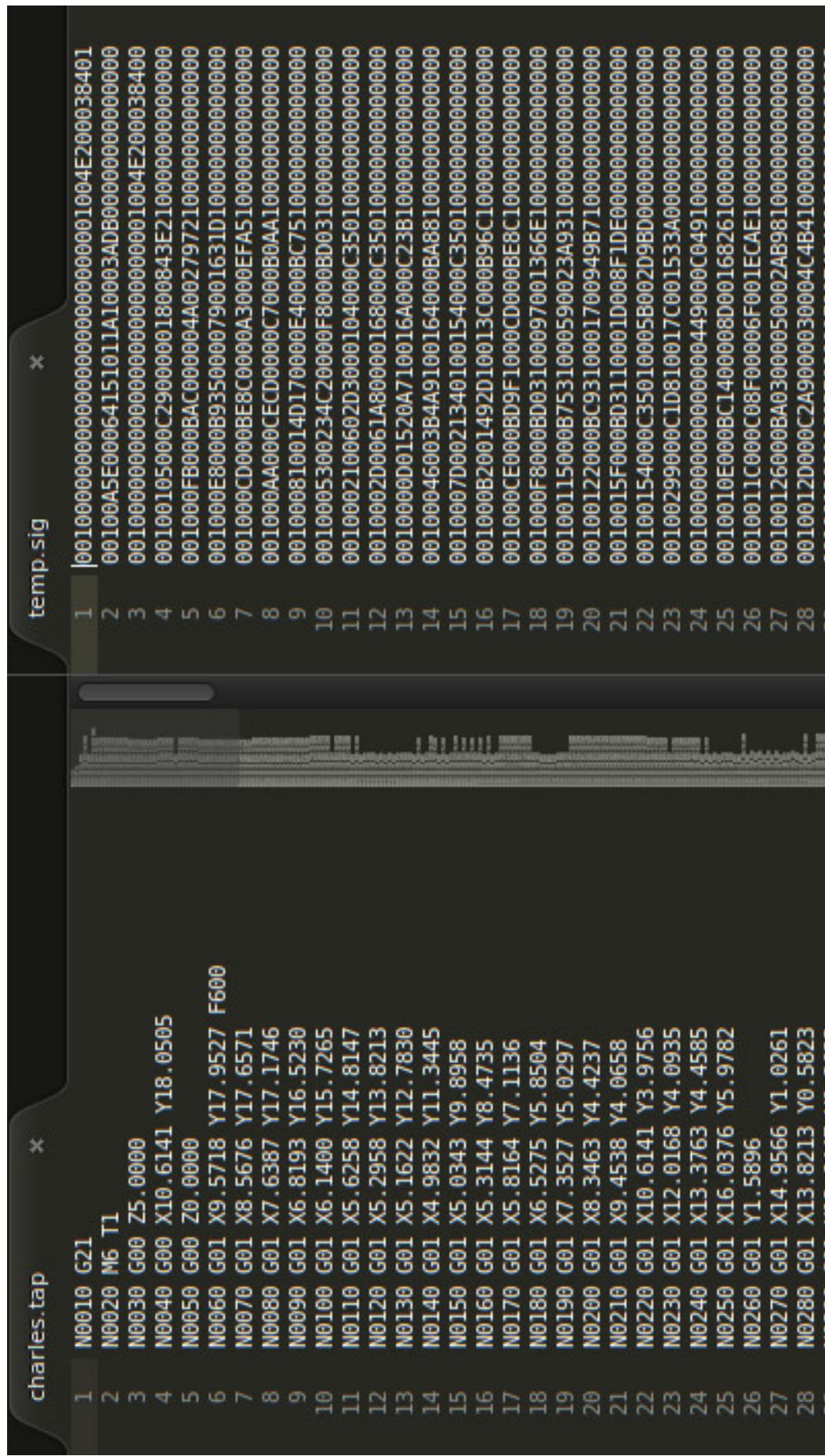
Bingol and Krishnamurty (2018) developed the *NURBS-Python* library, a scripted, object-oriented, open-source pure Python library without external dependencies. This open-source library availability provides access for researchers to develop NURBS applications. It is provided at the link [Python-NURBS Website]

Julia NURBS code is provided at the link [Julia NURBS library at Github] since Feb 20, 2019. Octave-NURBS is a collection of routines for the creation, and manipulation of Non-Uniform Rational B-Splines (NURBS), the latest release was on 2021-03-29. It is provided at link [Octave NURBS Website].

Note that, NURBS G-Code will not be in the scope of this research. In this thesis, to drive the CNC machine, the approach is to adopt the second-order approximations of Taylor's expansion, in an algorithm where the feedrate is calculated based on the value of chord-error and machine dynamic variables. The end objective is to constrain both chord-error and feedrate. For each point on the curve, the interpolation algorithm is computed in an iterative and recursive manner, back and forth, until it reaches a specified convergence criteria.

The interpolation algorithm in this work also generates a RS274D/NGC G-Code for each curve that later can be simulated offline, or executed directly on the UMP CNC machine to verify its success.

Figure 2.11: G-code file (left) and Signal file (right) for FPGA by Teh:2016



Chapter 3

METHODOLOGY

3.1 Parametric Curve Interpolation

This work involves the realtime interpolation of parametric curves with chord-error and feedrate constraints. The steps required are as follows:

1. Obtain the mathematical equations of functions describing the parametric curves.
2. Establish the relationship between the parametric curve, chord-error and machine feedrate in CNC terms.
3. Understand and formulate CNC machine dynamic and kinematic constraints.
4. Develop and execute a realtime algorithm that satisfies chord-error and feedrate constraints when run on the CNC machine.
5. Generate a RS274D NGC G-code for run simulations and offline execution of the algorithm.

3.2 Selection of parametric curves

Table 3.1: List of ten(10) selected parametric curves

1	Circle curve
2	Ellipse curve
3	Teardrop curve (illustrative curve)
4	Butterfly curve
5	Snailshell curve
6	Skewed-Astroid curve
7	Ribbon-10L curve
8	Ribbon-100L curve (10 times scale up of Ribbon10L curve)
9	AstEpi curve (Astroid and Epicycloid curves combined)
10	SnaHyp curve (Snailshell and Hypotrochoid curves combined)

The parametric curve in this work is a 2-dimensional curve of the type $C(x(u), y(u), z(u))$ where $z(u)$ is zero for all values of u .

All of the selected curves are at least C^2 continuous, meaning the second order derivative of $C(u)$ with respect to u must exist. It can be a constant but non zero for the range entire u range.

The selection of various parametric curves is aimed at obtaining curves with different features, shapes and dimensions. The curve characteristics cover variations, for example, in size of x and y dimensions, origin or not-origin centered, closed or open ended curves, number of loops, convex or concave segments, sharp or smooth turns, and reflection symmetry about the x and y axes.

3.3 Computing challenges in the algorithm

The main objective of selecting various complexities of parametric curves is to ensure that a single interpolation algorithm to be constructed can handle those complexities in computation without fail. In addition, the various shapes, sizes and features will impose a wide range of computing challenges to the algorithm.

Computations must be accurate and efficient. For sufficient precision, computations are conducted using 64-bit machines, and variables are specified in IEEE 754 double-precision binary floating-point format. This gives a precision of up to 15 to 17 significant decimal digits.

For example, the machine-epsilon (technically known as macheeps) in this work is between $1.11(10)^{-16}$ and $2.22(10)^{-16}$. Any non-zero number below macheeps is treated by the computing machine as technically zero. This is one of the many challenges in the algorithm especially when computing with very small real numbers. Note that precision is the number of digits specified, while accuracy is the difference of a number from its true value.

As an example, a curve that is x-axis symmetrical but not origin-centered will require the algorithm to handle both x and y offsets internally and invisibly. This is an additional challenge.

In this work, the Teardrop is considered the illustrative curve that provides many features like symmetry, closed-loop, smooth convex arc segments, a sharp pointed edge and varying width. It has all of the important elements to be handled by the interpolation algorithm. This is already considered complicated in comparison to a standard perfect circle which only has two features, that is, a center point and a constant radius.

For example, in this work, a perfect circle with a radius of 79 was selected instead of 80, a number rounded to multiples of 10 (decimal). This prime number 79 (which can only be divided by one and itself), is expected to impose computing challenges mathematically. It is also expected that in computing the successive interpolated points on the circle, it will produce a constant value of 79 for the radius of curvature, $\rho(u)$, over the entire range of parameter u . This is one of the many verification checks of the algorithm.

The Ellipse curve was selected because the Ellipse is essentially like a circle, but elongated vertically to be slim and tall. This feature is just a simple deviation from the perfect roundness of a circle.

In the same manner, because the Snailshell curve spirals inward, the radius of curvature $\rho(u)$ must be gradually and continuously decreasing,

Specifically, the two parametric curves for Ribbon10L and Ribbon100L were deliberately sized to be 10 times of each other. This is to check the validity of the interpolation algorithm on scaling. This up-scaling must be handled correctly by the algorithm. Additionally, these two curves are open-ended. In both cases it must be ensured that the algorithm really stops at the correct final positions.

The Skewed-Astroid curve was selected to have the algorithm handle four cusps at its corners. Instead of having a standard Astroid curve that is origin-centered and symmetrical about both the x and y axes, the Astroid curve was deliberately skewed to have a tall shape and very sharp cusps. It is very important that the algorithm handles points at these cusps correctly.

Combinations of standard curves were made to add complexity to the interpolation algorithm. The "AstEpi curve" is a combination of Astroid and Epicycloid curves while the "SnaHyp curve", is a combination of Snailshell and Hypotrocoid curves.

3.4 Characteristics of selected curves

Except for the two curves Ribbon-10L and Ribbon-100L, all other parametric curves listed for $x(u)$ and $y(u)$ involve a single parameter u on the right hand side (RHS) of the equations. These two curves require the u parameter range be normalized to $u \in [0.0, 1.0]$. The normalization procedure ensures that the interpolation algorithm cater only a single parameter range $u \in [0.0, 1.0]$, which must be applicable to all selected curves.

For example, the normalization for curves introduces a new parameter t . Firstly, a mathematical function $t(u)$ transforms the parameter u to t . Finally, this parameter t is used in the functions $x(t)$ and $y(t)$. The function $t(u)$ is called the parameter normalization function. This transformation is required to maintain $u \in [0.0, 1.0]$ range.

It was mentioned earlier that parametric representation of curves and surfaces allows for axis independence, multiple-valued functions and, can have infinite derivatives.

If the component functions are defined in terms of trigonometric functions like sine and cosine, or as exponential functions, the derivatives are infinite due to the cyclic nature of these functions. Some of the parametric curves selected in this work have such features. A polynomial function does not have such features because the n_{th} derivative will be zero after the n_{th} order of the polynomial. This is the case of the Teardrop curve where the polynomial is at least of third order to ensure that the second derivative exists and is non zero.

3.5 Curve shapes and curve equations

1. The selected curve shapes and their respective parametric equations are shown in the ensuing pages. It is intentional to start each curve on a new page.
2. To avoid figure distortion, each curve is displayed in a square view. Importantly, in order to obtain the correct impression of a curve, the reader is advised to take note of the x and y axes ranges, since different curves have different overall sizes.
3. The ranges for the x and y axes for each plot may be different. As far as possible, the figures are placed at the center of the grid.
4. Some figures obtained from published sources may be blurry and not clear. This is due to the original itself being blurry.
5. A summary data table for the curve shapes, dimensions and characteristics of the ten(10) curves selected in this work is provided in Table [3.1], in landscape mode.

Figure 3.1: Table data summary curve shapes and dimensions

ITEM	Parametric Curve	Parametric curves and dimensional characteristics					X-axis reflection	Y-axis reflection	Open ended	Num Loops	Shape remarks
		Origin centered	Width (mm)	Height (mm)	X-range	Y-range					
1	Teardrop	No	28.8	37.5	(-14.4, +14.4)	(0.0, -37.5)	Non-symm	Symm	No	1	All points below y-axis
2	Butterfly	No	58.0	49.4	(-28.9, +29.1)	(-18.1, +31.3)	Non-symm	Symm	No	7	Multiple size loops
3	Ellipse	Yes	22.0	102.0	(-11.0, +11.0)	(-51.0, +51.0)	Symm	Symm	No	1	Tall and slim
4	SkewedAstroid	Yes	80.0	200.0	(-40.0, +40.0)	(-100.0, +100.0)	Symm	Symm	No	0	4 cusps 4 concave curves
5	Circle	Yes	158.0	158.0	(-79.0, +79.0)	(-79.0, +79.0)	Symm	Symm	No	1	Radius set 79.0 not 80.0
6	AstEpi	Yes	117.2	117.2	(-58.6, +58.6)	(-58.6, +58.6)	Non-symm	Non-symm	No	3	All convex curves
7	Snailshell	Yes	32.7	40.9	(-10.8, +22.0)	(-15.9, +25.0)	Non-symm	Non-symm	Yes	0	smooth convex curves
8	SnaHyp	Yes	129.3	133.0	(-55.6, +73.7)	(-67.3, +133.0)	Non-symm	Non-symm	Yes	0	1 concave curve rest convex
9	Ribbon10L	No	4.0	4.0	(0.0, +4.0)	(+1.0, +5.0)	Non-symm	Non-symm	Yes	1	Very small curve size
10	Ribbon100L	No	40.0	40.0	(0.0, +40.0)	(+10.0, +50.0)	Non-symm	Non-symm	Yes	1	10 times scaleup Ribbin10L

3.6 Links to parametric curves

The following links will jump directly to the figures and their parametric equations.

1. Teardrop curve Fig [3.2]
2. Teardrop comparison: Zhong et. al.(2018) Fig [3.3]
3. Butterfly curve Fig [3.4]
4. Butterfly NURBS comparison: Ni et. al.(2018) Fig [3.5]
5. Butterfly NURBS comparison: Hu et. al.(2023) Fig [3.6]
6. Ellipse curve Fig [3.7]
7. SkewedAstroid curve Fig [3.8]
8. Circle Fig [3.9]
9. AstEpi curve Fig [3.10]
10. Snailshell curve Fig [3.11]
11. SnaHyp curve Fig [3.12]
12. Ribbon10L curve Fig [3.13]
13. Ribbon100L curve Fig [3.14]
14. Ribbon B-Splines comparison: Zhong-et-al(2018) Fig [3.15]

Figure 3.2: Teardrop shape profile

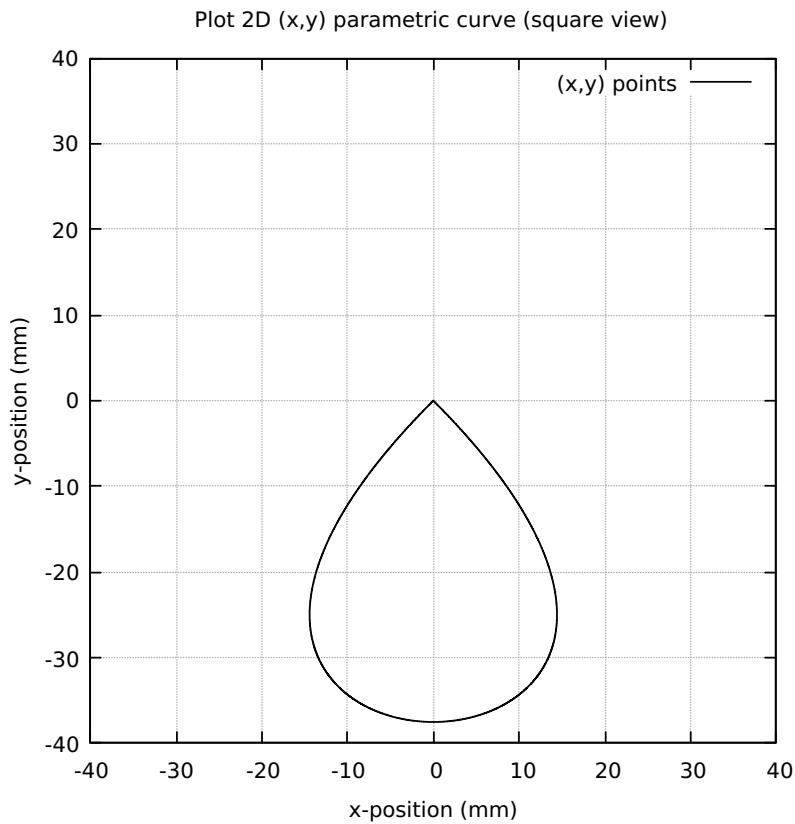
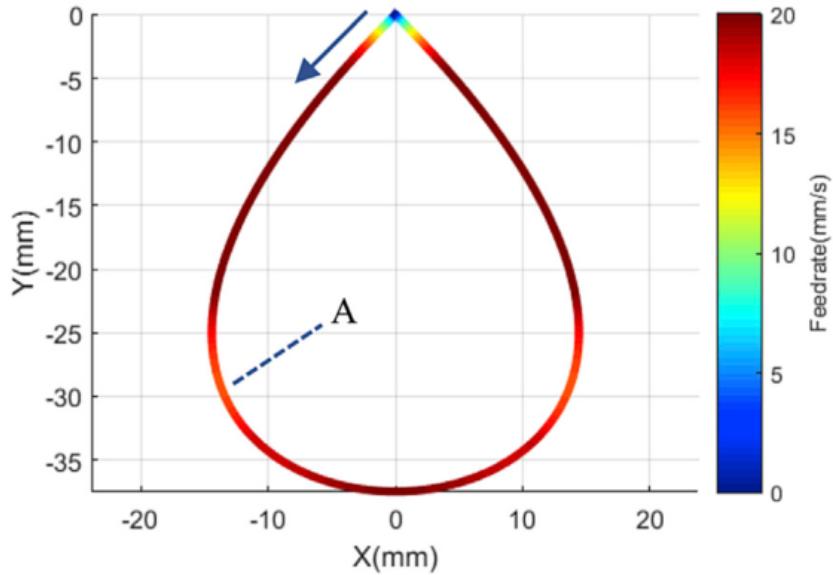


Table 3.2: Teardrop curve parametric equation

$$\begin{aligned}x(u) &= -150u + 450u^2 - 300u^3 \\y(u) &= -150u + 150u^2 \\u &\in [0.0, 1.0]\end{aligned}$$

Figure 3.3: Teardrop comparison Zhong et. al. (2018)



This figure by Zhong et. al. (2018) above is identical to the one used in this work because the same parametric equation was used. Even though the interpolation method is different, the results are similar but not identical. The flowchart in their work is complex and complicated to understand. As a comparison, the flowchart and algorithm in this work is simpler, and computations are fully functionalized and modularized.

Notice that this Teardrop figure by Zhong et. al. (2018) above is plotted on a rectangular grid (x and y intervals are not the same size), instead of a square grid done in this work shown in Fig [3.2]. This rectangular grid gives a false impression of a wider bottom for the Teardrop, that actually is not the case.

In addition, the Teardrop figure by Zhong is not presented in an overall grid that is square and origin-centered. It does not provide information on the placement of the Teardrop location relative to the origin. This origin-centered presentation provides information regarding the x and y axes reflection symmetry for the particular curve.

Figure 3.4: Butterfly shape profile

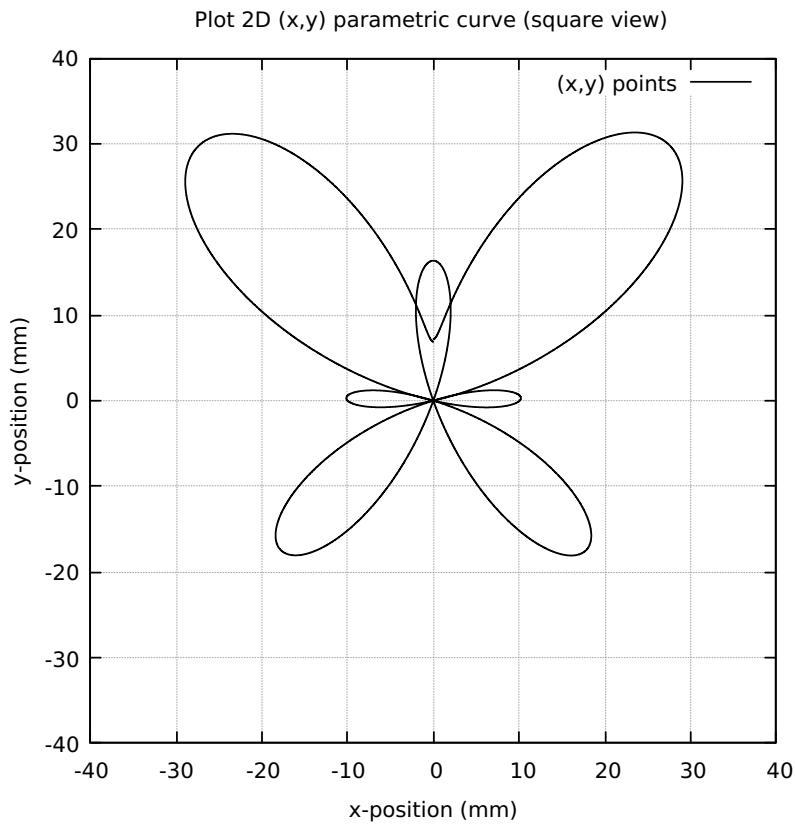


Table 3.3: Butterfly curve parametric equation

$$\begin{aligned}x(u) &= \sin(2\pi u) [e^{\cos(2\pi u)} - 2\cos(8\pi u) - (\sin(2\pi u/12))^5] \\y(u) &= \cos(2\pi u) [e^{\cos(2\pi u)} - 2\cos(8\pi u) - (\sin(2\pi u/12))^5] \\u &\in [0.0, 1.0]\end{aligned}$$

Figure 3.5: Butterfly comparison Ni et. al. (2018) NURBS

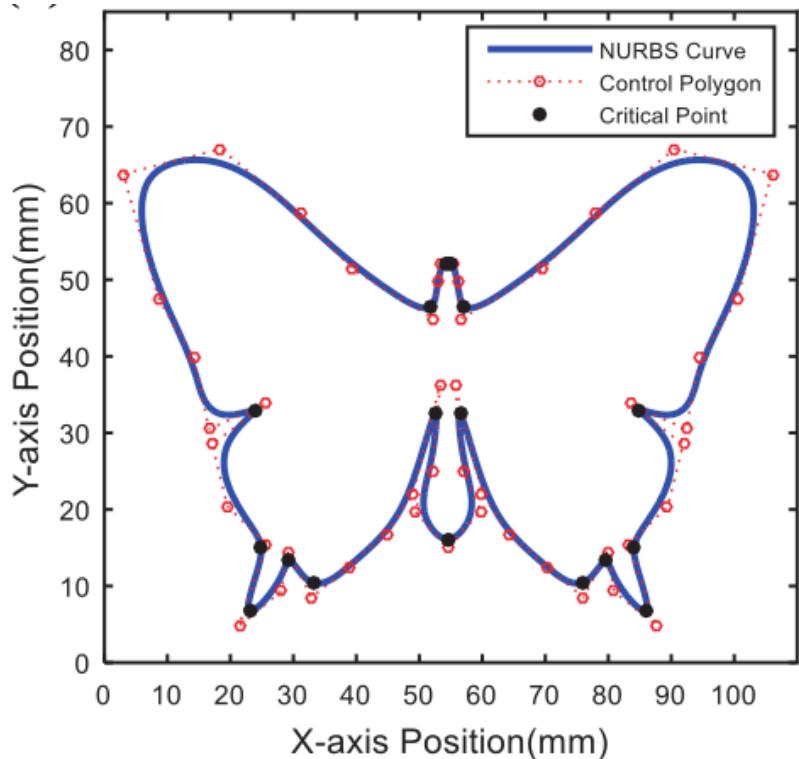


Figure 3.6: Butterfly comparison Hu et. al. (2023) NURBS

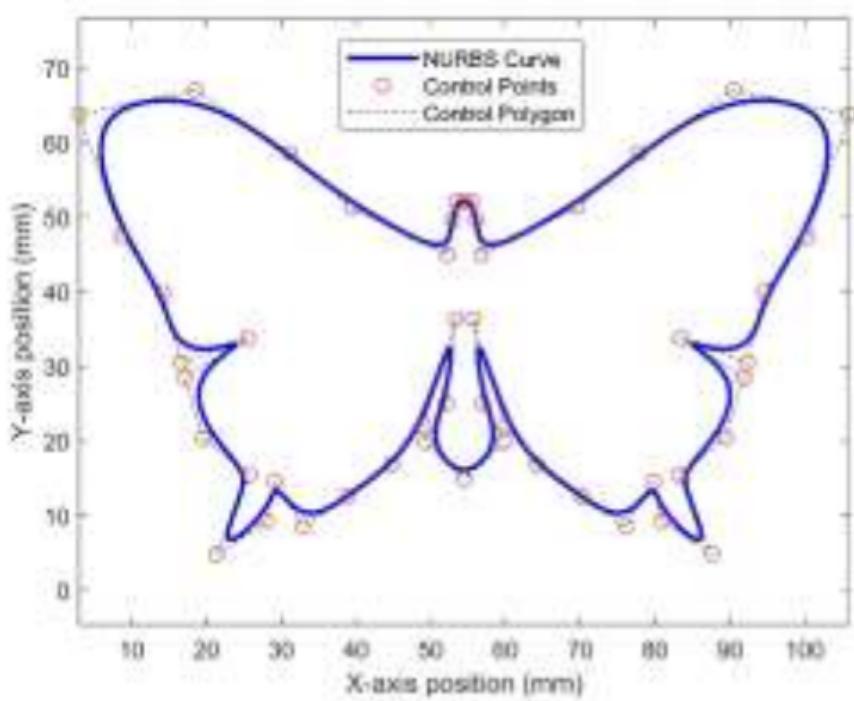


Figure 3.7: Ellipse shape profile

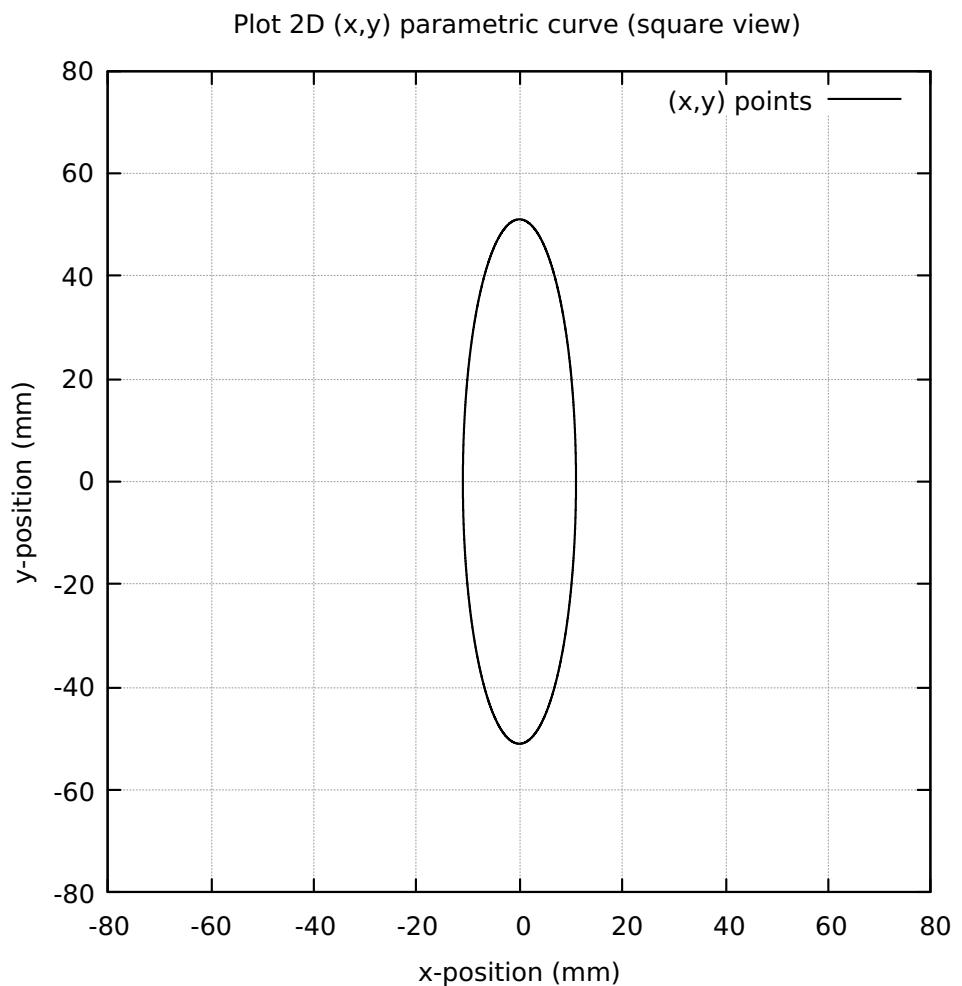


Table 3.4: Ellipse curve parametric equation

$$\begin{aligned}x(u) &= 11 \sin(2\pi u) \\y(u) &= 51 \cos(2\pi u) \\&\in [0.0, 1.0]\end{aligned}$$

Figure 3.8: SkewedAstroid shape profile

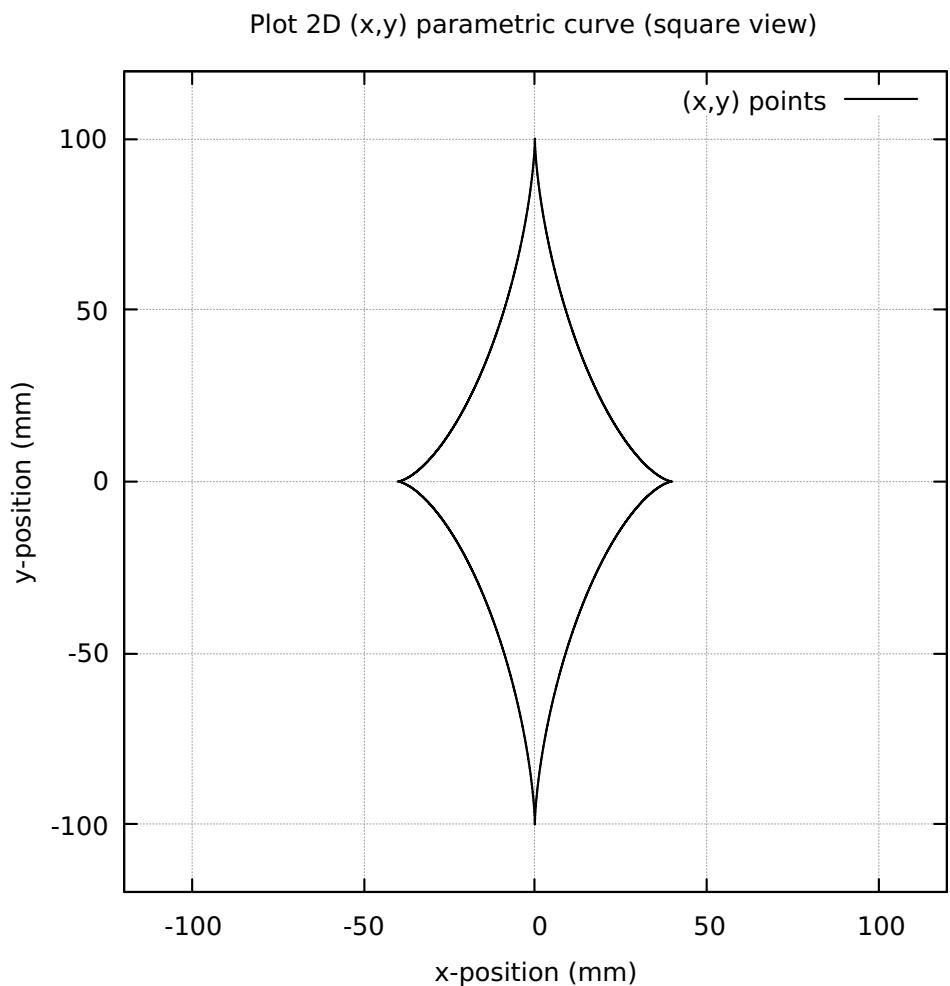


Table 3.5: SkewedAstroid curve parametric equation

$$\begin{aligned}x(u) &= 40[\sin(2\pi u)]^3 \\y(u) &= 100[\cos(2\pi u)]^3 \\u &\in [0.0, 1.0]\end{aligned}$$

Figure 3.9: Circle shape profile

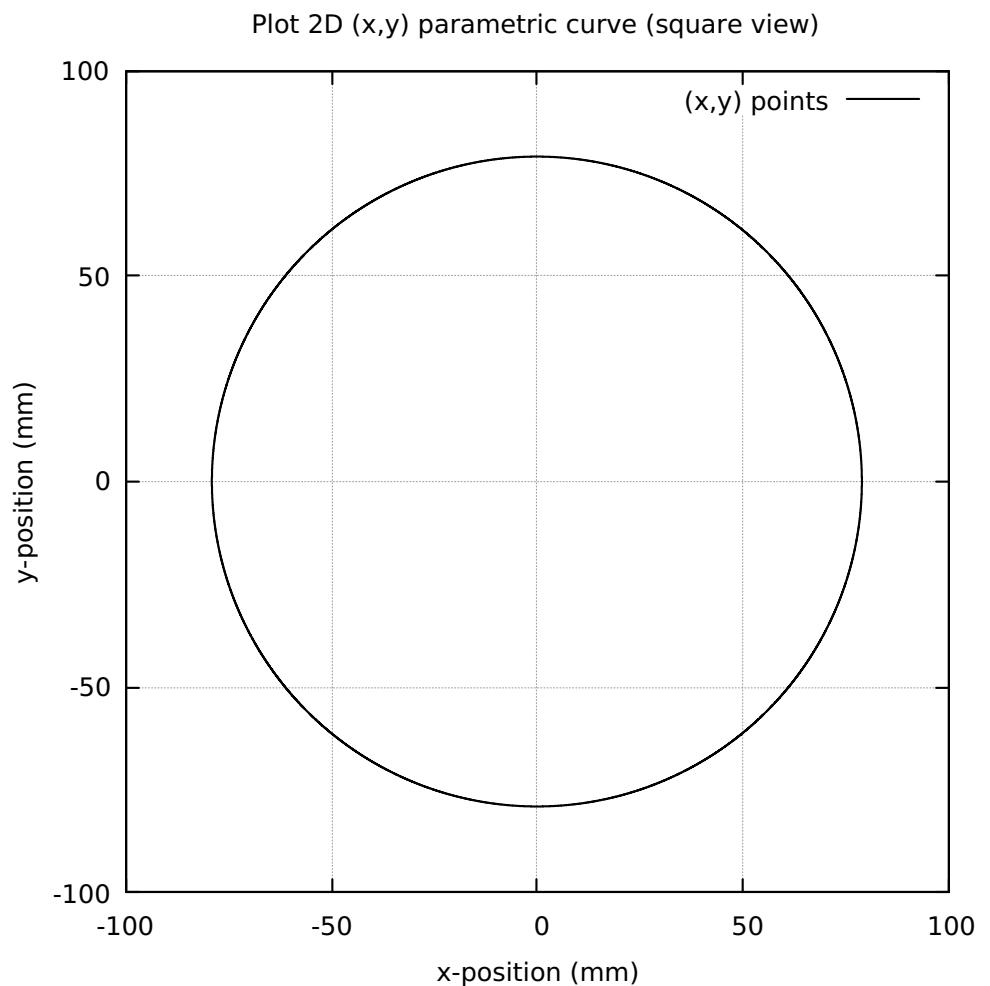


Table 3.6: Circle parametric equation

$$\begin{aligned}x(u) &= 79 \sin(2\pi u) \\y(u) &= 79 \cos(2\pi u) \\u &\in [0.0, 1.0]\end{aligned}$$

Figure 3.10: AstEpi shape profile

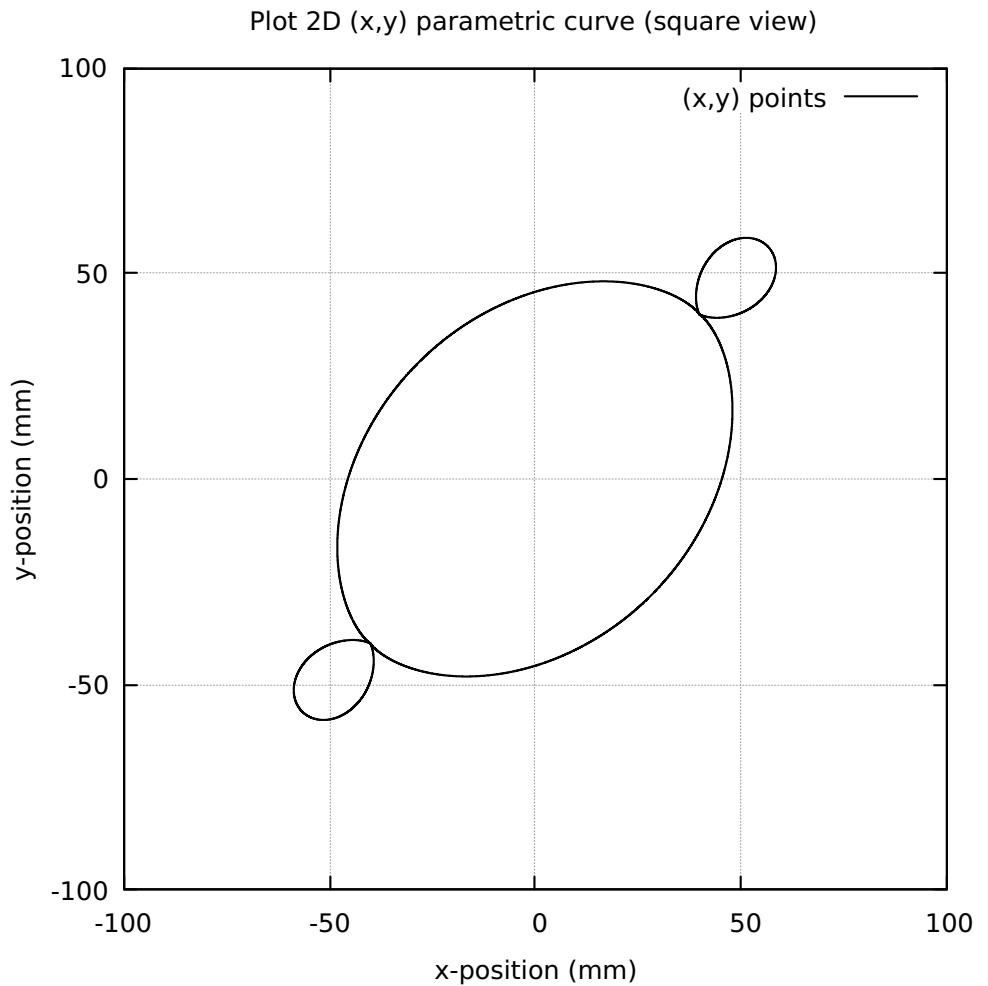


Table 3.7: AstEpi curve parametric equation

$$\begin{aligned}tvtiny &= 0.0000000001 \\x(u) &= 40[\sin(2\pi u)]^3 + 50 \cos(2\pi u + tvtiny) - 10 \cos(10\pi u - tvtiny) \\y(u) &= 40[\cos(2\pi u)]^3 + 50 \sin(2\pi u + tvtiny) - 10 \sin(10\pi u - tvtiny) \\u &\in [0.0, 1.0]\end{aligned}$$

Figure 3.11: Snailshell shape profile

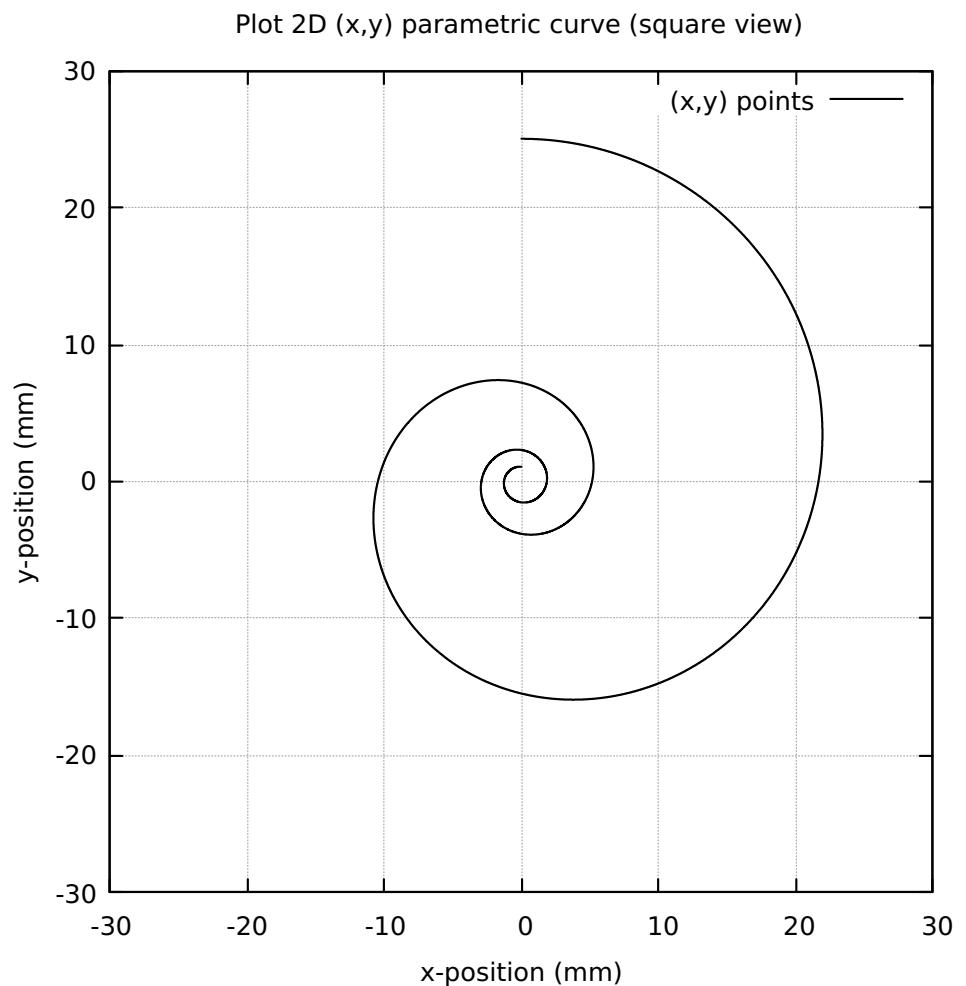


Table 3.8: Snailshell curve parametric equation

$$\begin{aligned}x(u) &= 100 \sin(6\pi u) / [9(\pi u)^2 + 4] \\y(u) &= 100 \cos(6\pi u) / [9(\pi u)^2 + 4] \\u &\in [0.0, 1.0]\end{aligned}$$

Figure 3.12: SnaHyp shape profile

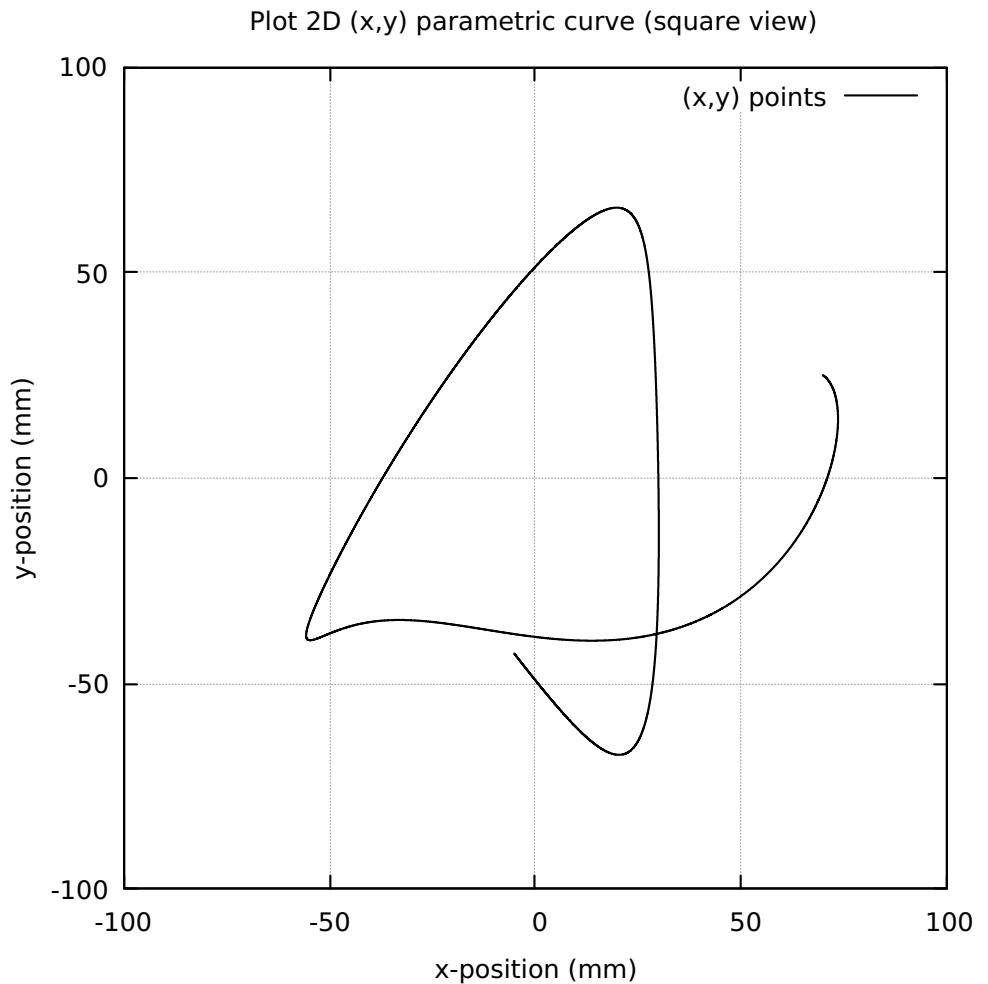


Table 3.9: SnaHyp curve parametric equation

$$\begin{aligned}
 xsna(u) &= [4 \sin(8\pi u)]/[16(\pi u)^2 + 4] \\
 xhyp(u) &= [2 \cos(4\pi u) + 5 \cos(8\pi u/3)] \\
 x(u) &= 10[xsna(u) + xhyp(u)] \\
 ysna(u) &= [10 \cos(8\pi u)]/[16(\pi u)^2 + 4] \\
 yhyp(u) &= [2 \sin(8\pi u) - 5 \sin(8\pi u/3)] \\
 y(u) &= 10[ysna(u) + yhyp(u)] \\
 u &\in [0.0, 1.0]
 \end{aligned}$$

Figure 3.13: Ribbon10L shape profile

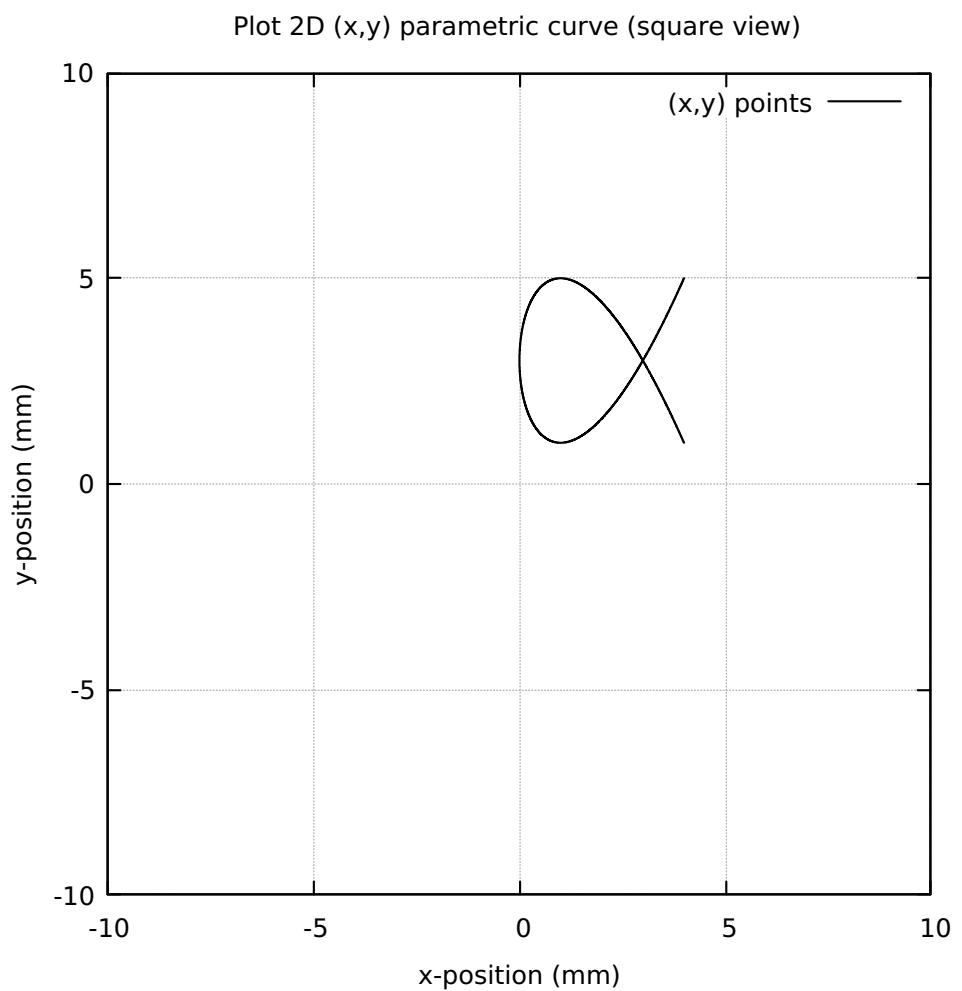


Table 3.10: Ribbon10L curve parametric equation

$$\begin{aligned}t(u) &= 4(u - 0.50) \\x(u) &= t^2 \\y(u) &= t^3 - 3t + 3 \\u &\in [0.0, 1.0]\end{aligned}$$

Figure 3.14: Ribbon100L shape profile

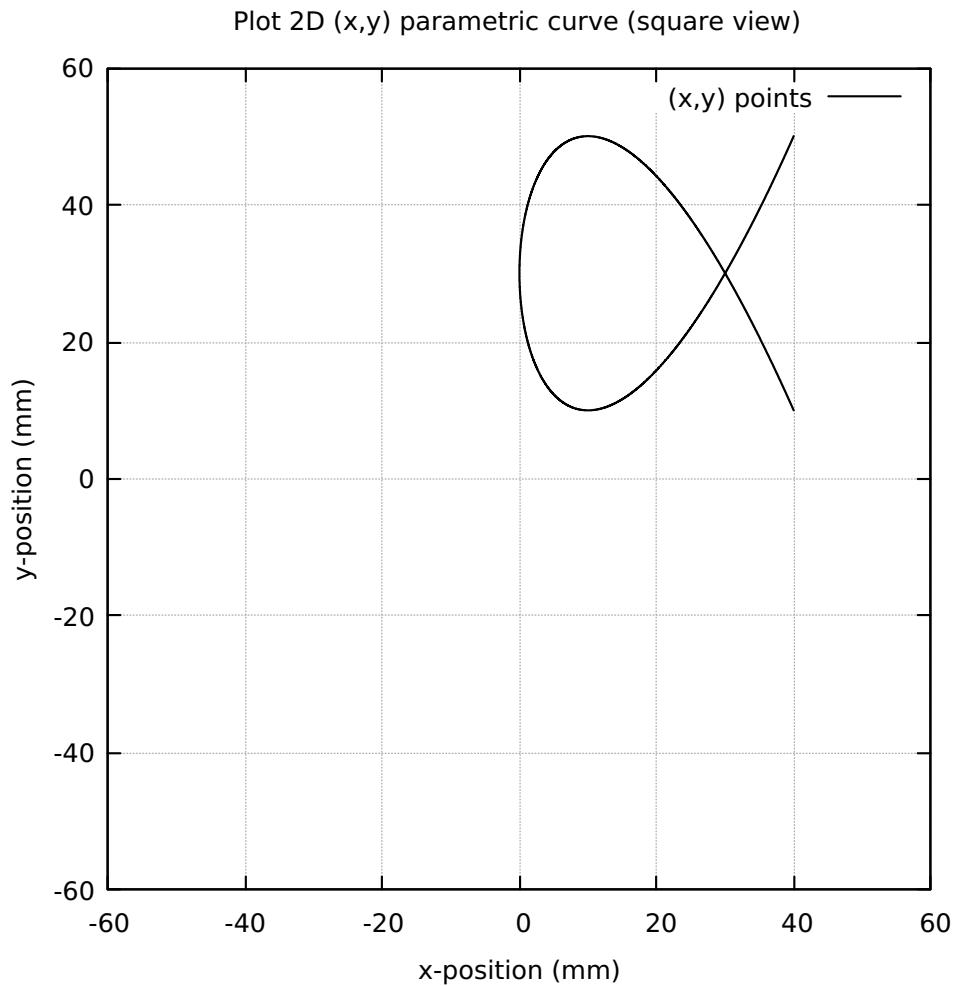
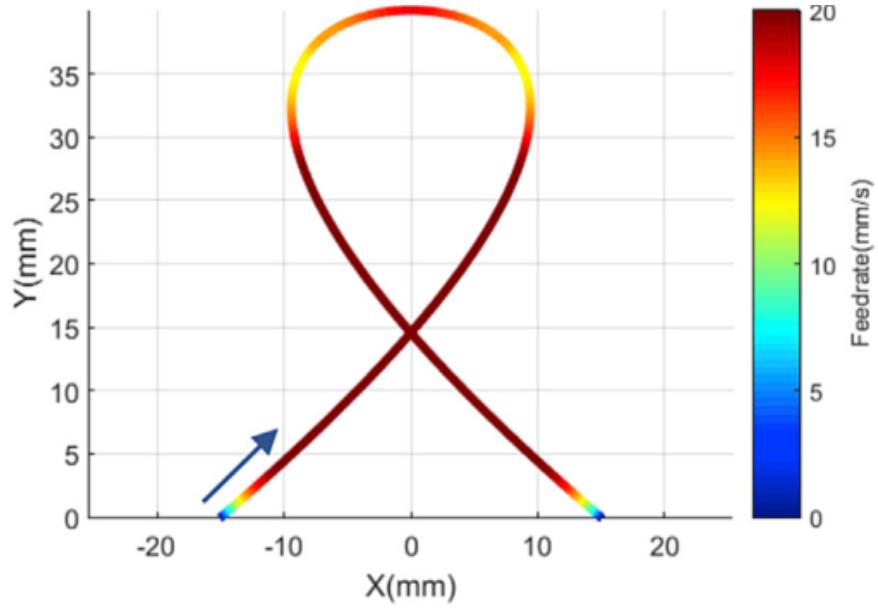


Table 3.11: Ribbon100L curve parametric equation

$$\begin{aligned}t(u) &= 4(u - 0.50) \\x(u) &= 10t^2 \\y(u) &= 10t^3 - 30t + 30 \\u &\in [0.0, 1.0]\end{aligned}$$

Figure 3.15: Ribbon comparison Zhong et. al. (2018) B-Splines



Compared to this work which uses the standard third degree polynomial for the Ribbon curve, the representation by Zhong et al. (2018) is a NURBS type, 3rd degree B-Spline with control points

$$\{P_i\} = \{(-15, 0), (20, 30), (0, 50), (-20, 30), (15, 0)\}$$

and knot vectors $U = \{0, 0, 0, 0, 0.5, 1, 1, 1, 1\}$.

NURBS or Non-Uniform Rational B-Splines are a generalization of B-Splines. Note that NURBS can create smooth and precise shapes that can be scaled, rotated, and deformed without losing quality. Splines are simpler curves that are defined by a set of control points and a degree of smoothness.

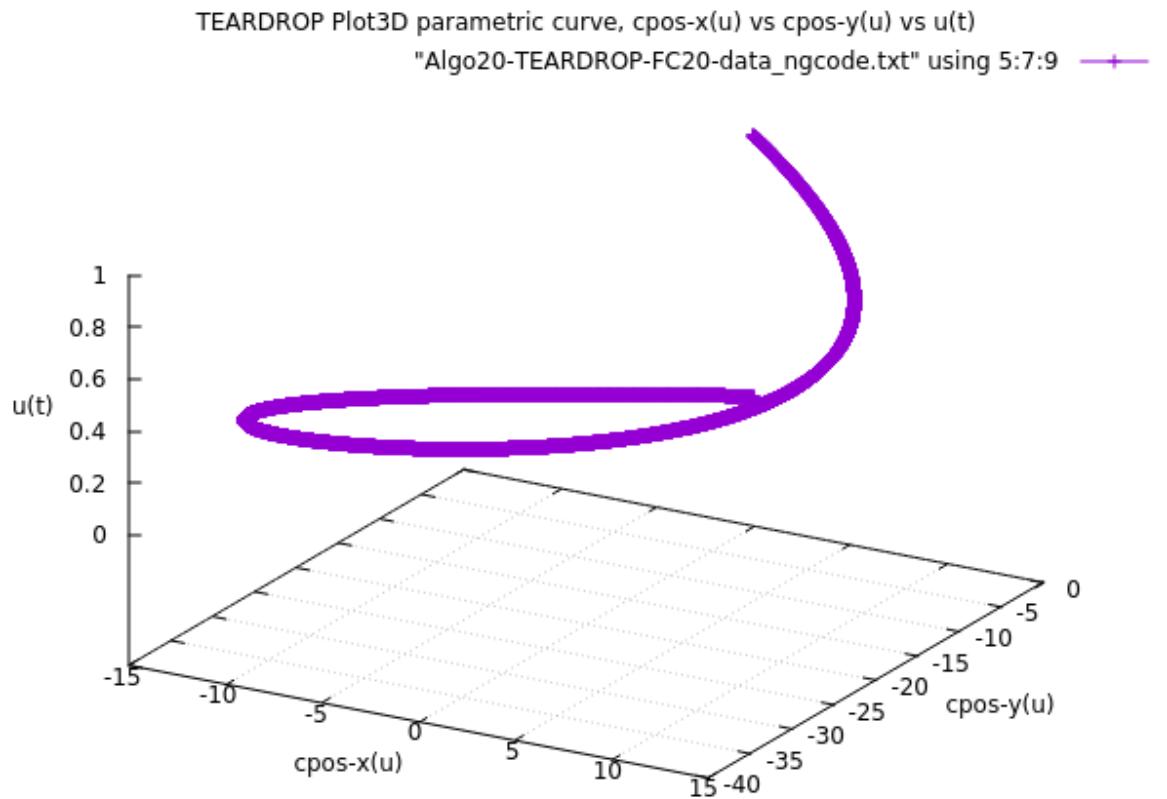


Figure 3.16: Teardrop 3D plot $x(u)$ vs $y(u)$ vs $u(t)$

3.7 Direction of travel in parametric curves

With a 3D plot of the Teardrop curve, direction of traversal of the u point can be easily visualized. It begins from $u = 0$ to $u = 1$. Upon collapsing the $u(t)$ axis, it becomes a 2D Teardrop closed curve again. This means parametric interpolations provide direction of traversal for curves and surfaces through the parameter $u(t)$.

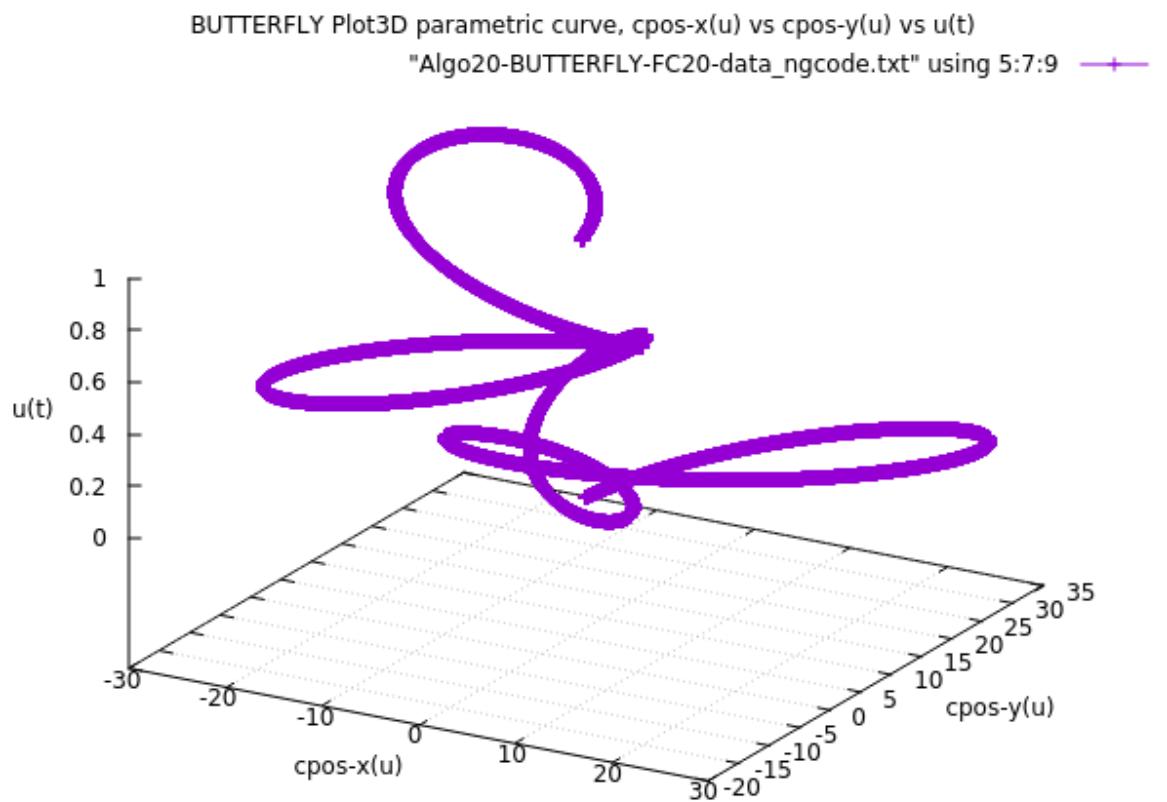
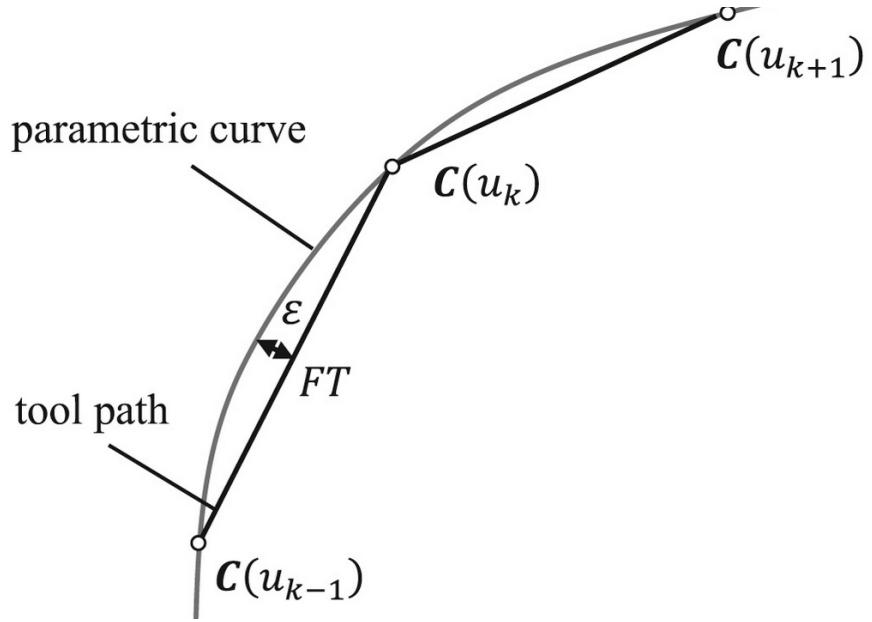


Figure 3.17: Butterfly 3D plot $x(u)$ vs $y(u)$ vs $u(t)$

The 3D plot for the Butterfly looks messy but the principle is the same. When the $u(t)$ axis collapses, it becomes a 2D Butterfly closed curve again.

Figure 3.18: Chord-error ϵ , Feedrate F, and Interpolation time T



3.8 Chord-error concept

The relationship between the parametric curve, chord-error and machine feedrate is illustrated in the Fig [3.18] above. (Source: Zhong et al. (2018)) It shows 3 successive interpolated points $C(u_{k-1})$, $C(u_k)$ and $C(u_{k+1})$ as the parameter is incremented from u_{k-1} to u_k and u_{k+1} along the parametric curve.

The chord is defined as the line connecting two points on the parametric curve. The chord-error ϵ is the maximum distance of a line from the curve perpendicular to the chord. If F is the feedrate (mm/s) and T is the interpolation period (s), that is, time duration to move from one point to the next, then the product of feedrate with interpolation time gives the length of the chord (FT).

The calculation method for the chord-error (ϵ) between two points on the parametric curve is described in section Chord-error $eps(u)$ [3.14].

3.9 Chord-error minimization

A decrease in the chord length (FT), will cause a decrease the chord-error (ϵ). This is favorable for accuracy since the toolpath follows closer to the path trajectory of the parametric curve. However, decreasing the chord-length causes an increase in the number of interpolated points for the same length of the curve. Since the interpolation time for a single step is a constant, this will also increase the total machining time, a feature that is not favorable.

3.10 Feedrate maximization

An increase in feedrate (F) of the CNC machining tool, will decrease the total machining time. This is favourable for faster completion of machining for the same length of the curve. However, the increase in feedrate will cause a larger chord length (FT), thus a larger chord-error (ϵ).

3.11 Chord-error and feedrate constraints

The approach in this work is to establish a combined chord-error and feedrate constraint. It is based on the following:

1. A strict maximum chord-error value is set as the error-tolerance. Every move in the interpolated point-to-point traversal must not exceed this error-tolerance.
2. Every move in the interpolated point-to-point traversal must not exceed the calculated feedrate limit for that particular point.
3. The feedrate limit at any particular point is calculated based on a combined geometric, dynamic and kinematic factors of the particular CNC machine.

3.12 Brief on algorithm strategy and design

In This-Work, the realtime parametric curve algorithm will be designed to iterate every point-to-point, as step move, such that the chord-error is below error-tolerance and the current feedrate is close but below the feedrate limit, that is, before the next move is taken. This is the criteria of iterative convergence. The cycle repeats until the point-to-point traversal of the entire parametric curve is completed.

Without exception, the flowcharts in This-Work are all of the type, "one way in and one way out". This strategy or paradigm is called "structured programming". Every functional computation unit implemented in this realtime interpolation algorithm comply with this structured programming strategy. In addition, various reports were generated, written to text files, to capture every variable change and transaction during the execution of the algorithm. This facilitates very easy search, analysis and debugging.

About structured programming algorithms, the following two(2) figures in the next section are basic examples of structured programming algorithms. Fig[3.19] and Fig[3.20] show the basic sequence and iterative loop. Both examples exhibit features of program flows in a "one way in and one way out" pattern.

About non-structured programming algorithms, the next two(2) figures demonstrate its features. The program flow in this case is difficult to track and so not easy to comprehend. Fig[3.21] shows overlapping flows in functional units "B" and "X" nodes, while Fig[3.22], is considered non-structured programming because there is "one-way-in but four-ways-out".

Figure 3.19: Example-1-Structured Programming Design (1-in, 1-out)

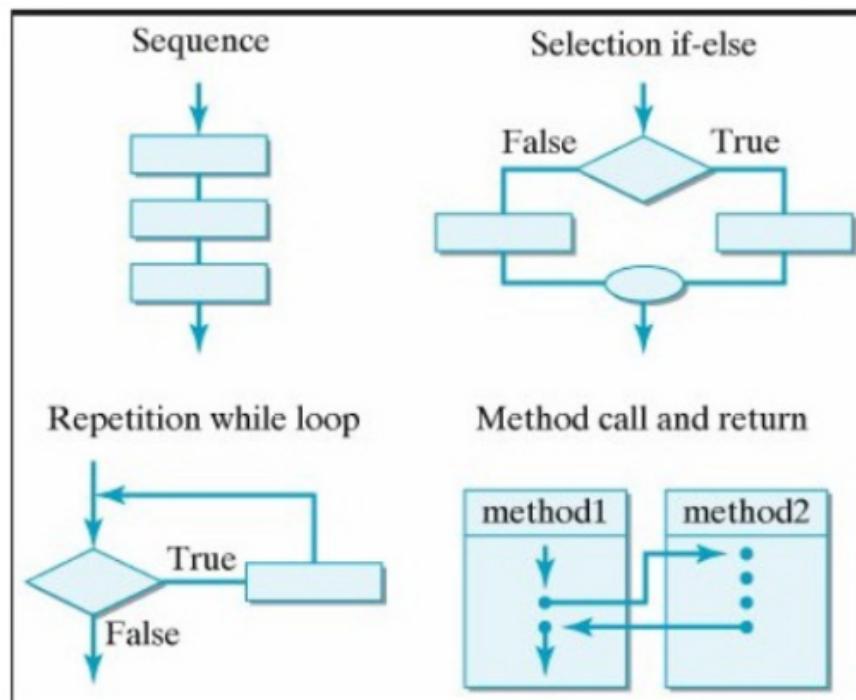


Figure 3.20: Example-2-Structured Programming Design (1-in, 1-out)

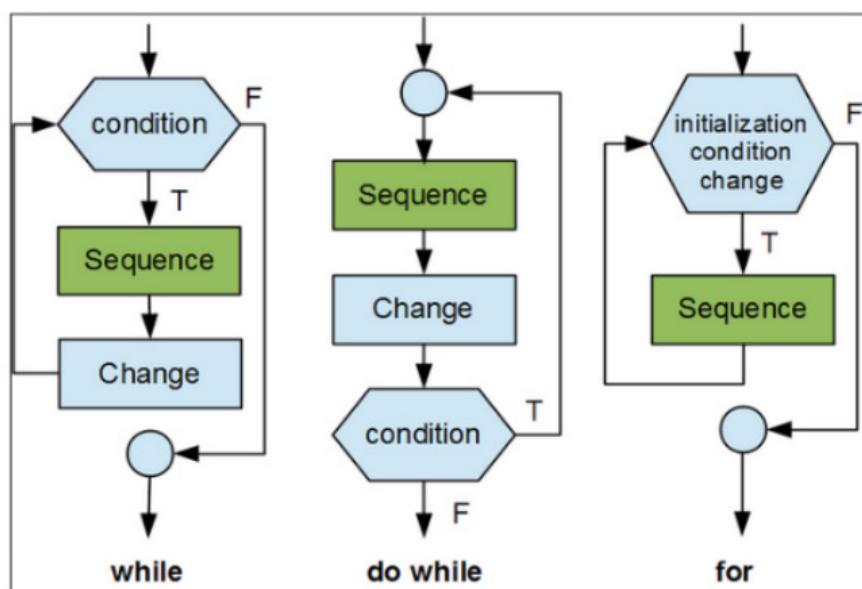


Figure 3.21: Example-3-Non-Structured Programming Design, (S=Start, E=Exit)

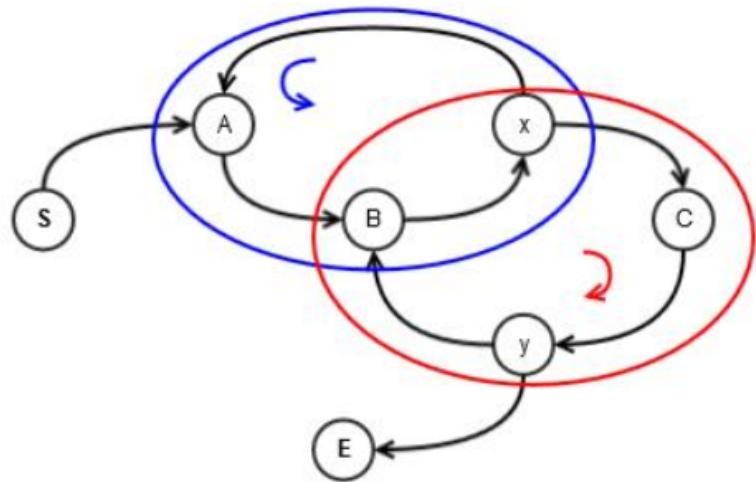
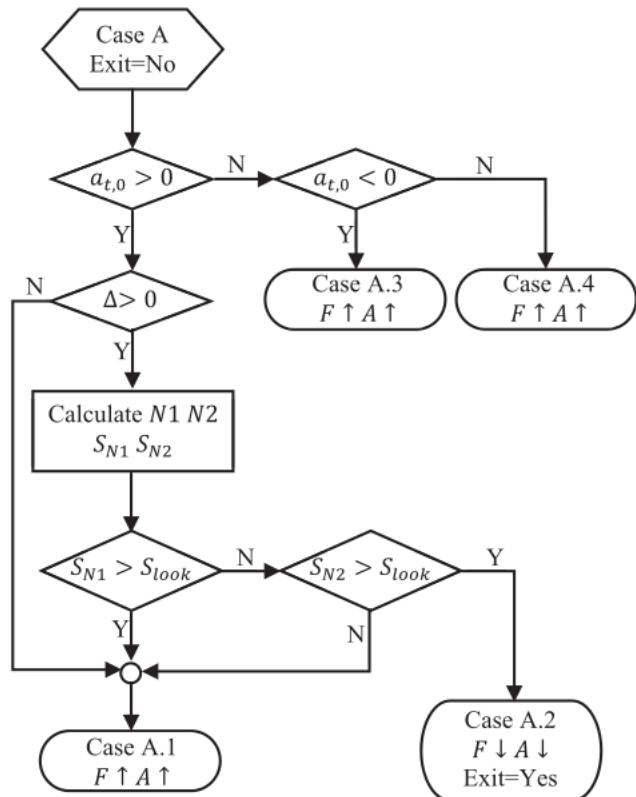


Figure 3.22: Example-4-Non-Structured Programming Design (1-in, 4-outs)



3.13 Radius of Curvature rho(u)

The radius of curvature (R or ρ) is important in inspection of curves and surfaces. The value of ρ is the reciprocal of curvature, K . So $\rho = (1/K)$. For a curve, $\rho(u)$ equals the radius of circular arc which best approximates the curve at that point u . The curvature K , is the amount by which a curve deviates from being a straight line, or a surface deviates from being a plane (not flat).

Note that the radius of curvature $\rho(u)$ is constant for a circle over all applicable values of parameter u . Essentially, the value of $\rho(u)$ is just the radius of the circle itself.

Consider a series of concentric circles with the radii increasing from zero to some large number. Intuitively, an increasing $\rho(u)$ value means the curvature opens to a larger and larger circle radius. On the contrary, a decreasing $\rho(u)$ value means the curvature closes to a smaller and smaller circle radius.

When $\rho(u)$ becomes zero, the curve shrinks to a single point and there is no curvature or curving behaviour. When $\rho(u)$ value reaches infinity, it becomes a straight line and there is also no curving behaviour. This is an important general rule when visually inspecting curves and surfaces.

If the curve is given parametrically by functions $x(u)$ and $y(u)$, then from standard calculus and analytic geometry, the exact derivation for radius of curvature is given by:

$$R(u) = \rho(u) = \left| \frac{\text{numerator}(u)}{\text{denominator}(u)} \right|$$

$$K(u) = \text{curvature}(u) = \left| \frac{\text{denominator}(u)}{\text{numerator}(u)} \right|$$

where

$$\text{numerator}(u) = \left(\left(\frac{dx(u)}{du} \right)^2 + \left(\frac{dy(u)}{du} \right)^2 \right)^{3/2}$$

$$\text{denominator}(u) = \left(\frac{dx(u)}{du} \right) \left(\frac{d^2y(u)}{du^2} \right) - \left(\frac{d^2x(u)}{du^2} \right) \left(\frac{dy(u)}{du} \right)$$

3.14 Chord-error eps(u)

The chord-error (epsilon) is the maximum length of the line perpendicular to the chord from a point on the arc segment. Based on extensive computer simulations, it was found that the perpendicular line starting from the mid-point on the chord that intersects the arc segment gives the maximum length. The algorithm for calculating $\text{eps}(u)$ is as follows:

1. From the current u -point giving an $C(x(u), y(u))$ point on the parametric curve, determine the $C(x(u + next), y(u + next))$ point value on the curve. This is just the next interpolated point on the curve.

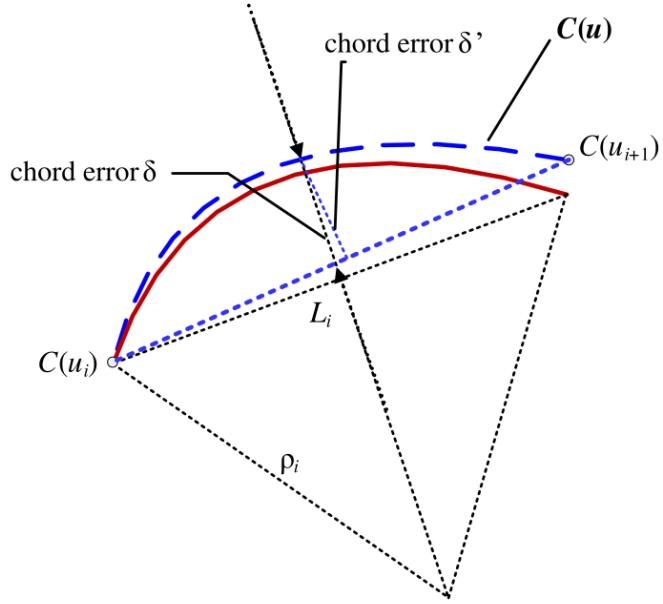


Figure 3.23: Chord-error between two interpolated points

2. Construct a chord line between the points $C(x(u), y(u))$ and $C(x(u + next), y(u + next))$. Calculate the slope of this chord line. Call it *chordslope*.
3. From geometry, any line that is perpendicular to this chord will have a slope of *perplineslope* = $(-1/\text{chordslope})$.
4. Find the mid-point coordinates x_{mid} and y_{mid} on this chord line.
5. From the known *chordslope* and chord mid-point coordinates, generate a linear equation perpendicular to this line. Call it the *perpline*.
6. Find the intersection coordinates of the given parametric curve (arc segment) with the *perpline*. Call these coordinates x_{int} and y_{int} .
7. The calculated value of chord-error $\text{eps}(u)$ is the linear length between the points (x_{mid}, y_{mid}) and (x_{int}, y_{int}) .
8. The linear length is calculated using the standard Pythagoras formula in Cartesian coordinates.

3.15 Feedrate or Velocity $\mathbf{V}(\mathbf{u})$

This section describes the derivation of feedrate or velocity ($V(u_i)$) from the radius of curvature ($rho(u_i)$) and chord-error ($eps(u_i)$). For a linear distance traveled (chord distance) between $C(u_i)$ and $C(u_{i+1})$, denoted by L_i , the velocity is given by

$$V(u_i) = \frac{L_i}{T_i}$$

where $T_i = (t_{i+1} - t_i)$ is the interpolation time. Since an arc approximates a section of the circle, the chord error $eps(u_i)$, is derived from geometry as a function of the radius of curvature $rho(u_i)$, approximately as follows:

$$eps(u_i) = (radius_to_arc) - (radius_to_center_of_chord)$$

$$eps(u_i) = rho(u_i) - \sqrt{rho(u_i)^2 - (L_i/2)^2}$$

Solving for L_i and replacing for L_i , the velocity is

$$V(u_i) = \frac{L_i}{T_i} = \frac{2}{T_i} \cdot \sqrt{rho(u_i)^2 - (rho(u_i) - eps(u_i))^2}$$

3.16 First order Taylor's approximation u_next(u)

Consider the parametric curve function $C(u)$, and the time function $u(t)$ as the curve parameter moves from $u(t_i) = u_i$ to $u(t_{i+1}) = u_{i+1}$. By using a Taylor's expansion, the approximation up to the first derivative is

$$u(t_{i+1}) = u(t_i) + (t_{i+1} - t_i) \cdot \left(\frac{du}{dt} \right) \Big|_{t=t_i} + \text{Higher_order_terms}$$

$$u_{i+1} = u_i + (t_{i+1} - t_i) \cdot \left(\frac{du}{dt} \right) \Big|_{t=t_i} + \text{Higher_order_terms}$$

The speed $V(u_i)$ with respect to time at u_i from the parametric curve $C(u)$ is

$$V(u_i) = \left(\frac{dC(u)}{du} \right) \Big|_{u=u_i} \cdot \left(\frac{du}{dt} \right) \Big|_{t=t_i}$$

Rearranging the equation

$$\left(\frac{du}{dt} \right) \Big|_{t=t_i} = \frac{V(u_i)}{\left(\frac{dC(u)}{du} \right) \Big|_{u=u_i}}$$

The Taylor's expansion now becomes

$$u_{i+1} = u_i + (t_{i+1} - t_i) \cdot \frac{V(u_i)}{\left(\frac{dC(u)}{du} \right) \Big|_{u=u_i}} + \text{Higher_order_terms}$$

Neglecting the higher order terms and with the interpolation time or step time defined as $T_i = (t_{i+1} - t_i)$

$$u_{i+1} = u_i + unext1(u)$$

$$u_{i+1} = u_i + \frac{(T_i) \cdot V(u_i)}{\left(\frac{dC(u)}{du} \right) \Big|_{u=u_i}}$$

Note that the denominator cannot be zero.

3.17 Second order Taylor's approximation u_next(u)

Define the interpolation time $T_i = (t_{i+1} - t_i)$

By using a Taylor's expansion, the approximation up to the second derivative is

$$u(t_{i+1}) = u(t_i) + (T_i) \cdot \left(\frac{du(t)}{dt} \right) \Big|_{t=t_i} + \frac{T_i^2}{2} \cdot \left(\frac{d^2u(t)}{dt^2} \right) \Big|_{t=t_i} + \text{Higher_order_terms}$$

Neglecting the higher order terms,

$$\begin{aligned} u(t_{i+1}) &= u(t_i) + (T_i) \cdot \left(\frac{du(t)}{dt} \right) \Big|_{t=t_i} + \frac{T_i^2}{2} \cdot \left(\frac{d^2u(t)}{dt^2} \right) \Big|_{t=t_i} \\ \left(\frac{d^2u(t)}{dt^2} \right) &= \frac{d \left(\frac{du(t)}{dt} \right)}{dt} = \frac{df(t)}{dt} \end{aligned}$$

Given arbitrary functions $f(t)$, $g(t)$ and $h(t)$, where:

$$f(t) = \frac{g(t)}{h(t)}$$

The Quotient Rule says that the derivative of a quotient is the denominator times the derivative of the numerator minus the numerator times the derivative of the denominator, all divided by the square of the denominator. Using the quotient rule for the derivative gives:

$$\frac{df(t)}{dt} = \frac{h(t) \cdot \left(\frac{dg(t)}{dt} \right) - g(t) \cdot \left(\frac{dh(t)}{dt} \right)}{(h(t))^2}$$

With the function $f(t)$ calculated previously

$$\begin{aligned} f(t) &= \left(\frac{du(t)}{dt} \right) = \frac{V(u_i)}{\left(\frac{dC(u)}{du} \right) \Big|_{u=u_i}} \\ \frac{df(t)}{dt} &= \left(\frac{d^2u(t)}{dt^2} \right) = \frac{d \left(\frac{g(t)/h(t)}{du} \right)}{dt} \end{aligned}$$

where

$$g(t) = V(u_i)$$

$$h(t) = \left(\frac{dC(u)}{du} \right) \Big|_{u=u_i}$$

Substitute $g(t)$ and $h(t)$ into the quotient rule equation and rearranging

$$\frac{df(t)}{dt} = \left(\frac{d^2u(t)}{dt^2} \right) = -(V(u_i))^2 \cdot \frac{A}{B}$$

$$\left(\frac{d^2u(t)}{dt^2} \right) = -(V(u_i))^2 \cdot \frac{\left| \left(\frac{d^2C(u)}{du^2} \right) \right|_{u=u_i}}{\left| \left(\frac{dC(u)}{du} \right) \right|_{u=u_i}^3}$$

where

$$A = \left| \left(\frac{d^2C(u)}{du^2} \right) \right|_{u=u_i}$$

$$B = \left| \left(\frac{dC(u)}{du} \right) \right|_{u=u_i}^3$$

The second order Taylor's expansion for the function $u(t)$ becomes

$$u(t_{i+1}) = u(t_i) + (T_i) \cdot \left(\frac{du}{dt} \right) \Big|_{t=t_i} + \frac{T_i^2}{2} \cdot \left(\frac{d^2u}{dt^2} \right) \Big|_{t=t_i}$$

$$u(t_{i+1}) = u(t_i) + \frac{(T_i \cdot V(u_i))}{\left| \left(\frac{dC(u)}{du} \right) \right|_{u=u_i}} - \frac{(T_i \cdot V(u_i))^2}{2} \cdot \frac{\left| \left(\frac{d^2C(u)}{du^2} \right) \right|_{u=u_i}}{\left| \left(\frac{dC(u)}{du} \right) \right|_{u=u_i}^3}$$

In software implementation,

$$u_next = u(t_{i+1}) - u(t_i)$$

$$u_next = \frac{(T_i \cdot V(u_i))}{\left| \left(\frac{dC(u)}{du} \right) \right|_{u=u_i}} - \frac{(T_i \cdot V(u_i))^2}{2} \cdot \frac{\left| \left(\frac{d^2C(u)}{du^2} \right) \right|_{u=u_i}}{\left| \left(\frac{dC(u)}{du} \right) \right|_{u=u_i}^3}$$

where

$$\left| \left(\frac{dC(u)}{du} \right) \right|_{u=u_i} = fxn_cvel_magn(u)$$

$$\left| \left(\frac{d^2C(u)}{du^2} \right) \right|_{u=u_i} = fxn_cacc_magn(u)$$

$$T_i = (t_{i+1} - t_i) = \text{implementation_time_selected}$$

$$V(u_i) = \text{velocity_command_or_feedrate_generated_by_some_function}$$

3.18 Next interpolation point calculation

In the realtime interpolation algorithm, the next interpolation point u_{next} is calculated based on Taylor's expansion as follows. Note that for u_{next} to be valid, the denominators cannot be zero.

First order Taylor's approximation

$$u_{\text{next}} = \frac{(T_i).V(u_i)}{\left(\frac{dC(u)}{du}\right)\Big|_{u=u_i}}$$

Second order Taylor's approximation

$$u_{\text{next}} = \frac{(T_i.V(u_i))}{\left(\frac{dC(u)}{du}\right)\Big|_{u=u_i}} - \frac{(T_i.V(u_i))^2}{2} \cdot \frac{\left|\left(\frac{d^2C(u)}{du^2}\right)\right|_{u=u_i}}{\left|\left(\frac{dC(u)}{du}\right)\right|_{u=u_i}^3}$$

The above are the two(2) equations to be used in software implementation. Notice the negative value of the second term in the Second order Taylor's approximation.

It is important to note that all of the derivatives (first and second order) are executed on the curve $C(u)$ with respect to parameter u . The time parameter t is no longer in the equation. Essentially, the Taylor's approximation used here "transforms" the u_{next} point in terms of u instead of t . The starting point is u being a "function of t ", that is, $u(t)$ ".

3.19 Feedrate limit calculations

The current feedrate limit is the minimum of four(4) separate feedrate limits denoted by $(fratelimit_1, fratelimit_2, fratelimit_3, fratelimit_4)$. The limits are based on geometrical and dynamical constraints described below.

$fratelimit_1$ = frate_command, a user specified feedrate constraint value

$fratelimit_2$ = dynamic constraint on machine maximum X-Y axial velocities

$fratelimit_3$ = geometric constraint on chord-error or contour accuracy

$fratelimit_4$ = dynamic constraint on machine maximum X-Y axial accelerations

The functional form for the current feedrate limit becomes:

$$\text{feedrate_limit} = \min(fratelimit_1, fratelimit_2, fratelimit_3, fratelimit_4)$$

The current feedrate limit at u is calculated as a function of the following variables.

Table 3.12: Feedrate limit function parameters

rt	= current runtime in seconds
u	= current u parameter value
u_{next}	= next interpolated u parameter value
FC	= user specified feedrate command (constant) value
T	= interpolation step time (constant 0.001) in seconds
ρ	= radius of curvature at current u value
ϵ	= chord-error (epsilon) at current u value
λ	= safety factor for acceleration (a constant, range 0.0 - 1.0)
$xVel_{max}$	= maximum allowable x-axis velocity of the machine
$yVel_{max}$	= maximum allowable y-axis velocity of the machine
$xAcc_{max}$	= maximum allowable x-axis acceleration of the machine
$yAcc_{max}$	= maximum allowable y-axis acceleration of the machine
α_{Vel}	= magnitude of x-velocity unit vector at current u
β_{Vel}	= magnitude of y-velocity unit vector at current u

It is important to note that the feedrate limit calculations for the Teardrop curve were conducted as a comparison to the work of Zhong et al. (2018). The calculations overall were different, but the resulting feedrate profiles were similar. This work however, produces a much smoother feedrate profile.

3.19.1 Feedrate Limit 1

Table 3.13: Calculation for *fratelimit_1*

$$fratelimit_1 = user_assigned_fixed_value(e.g.20)$$

The machining feedrate during correct machine operation should never exceed this user assigned feedrate value. Thus, *fratelimit_1* is the absolute maximum limit, and therefore, appropriately termed as the *feedrate_command*. For a perfect and ideal operation, the machine should be running at this feedrate value throughout the curve path. This should happen for a path that is exactly a straight line.

3.19.2 Feedrate Limit 2

Table 3.14: Calculation for *fratelimit_2*

$$\begin{aligned} alphaVel(u) &= \left| \frac{\left(\frac{dx(u)}{du} \right)}{\sqrt{\left(\frac{dx(u)}{du} \right)^2 + \left(\frac{dy(u)}{du} \right)^2}} \right| \\ betaVel(u) &= \left| \frac{\left(\frac{dy(u)}{du} \right)}{\sqrt{\left(\frac{dx(u)}{du} \right)^2 + \left(\frac{dy(u)}{du} \right)^2}} \right| \\ A &= xVel_{max}/alphaVel(u) \\ B &= yVel_{max}/betaVel(u) \\ fratelimit_2 &= minimum(A, B) \end{aligned}$$

The *fratelimit_2* is dependent on both geometrical constraints *alphaVel(u)* and *betaVel(u)*, as well as dynamical constraints, that is, the maximum allowable axial velocities of the machine as shown above ($xVel_{max}$ and $yVel_{max}$) The quantities *alphaVel(u)*, *betaVel(u)* are essentially the magnitudes of curve unit vectors in the x and y directions, respectively.

3.19.3 Feedrate Limit 3

Table 3.15: Calculation for `fratelimit_3`

$$\begin{aligned}
 numerator(u) &= \left(\left(\frac{dx(u)}{du} \right)^2 + \left(\frac{dy(u)}{du} \right)^2 \right)^{3/2} \\
 denominator(u) &= \left(\frac{dx(u)}{du} \right) \left(\frac{d^2y(u)}{du^2} \right) - \left(\frac{d^2x(u)}{du^2} \right) \left(\frac{dy(u)}{du} \right) \\
 rho &= numerator(u)/denominator(u) \\
 eps &= rho - \sqrt{rho^2 - (L/2)^2} \\
 fratelimit_3 &= (2/T) * \left| \sqrt{(2 * rho * eps - eps^2)} \right|
 \end{aligned}$$

The `fratelimit_3` is primarily dependent on curve geometry, as seen above involving radius of curvature $\rho(u)$ and chord-error $\epsilon(u)$. Both are dependent on the independent parameter u and the curve equations. For example, a perfect circle would give a constant radius of curvature $\rho(u)$ and the chord lengths would all be the same.

3.19.4 Feedrate Limit 4

Table 3.16: Calculation for *fratelimit_4*

$$\begin{aligned}
 \lambda &= \text{user_select_safety_factor}(0.0 \text{to} 1.0) \\
 C &= \sqrt{\frac{\lambda * \rho * xAcc_{max}}{|betaVel|}} \\
 D &= \sqrt{\frac{\lambda * \rho * yAcc_{max}}{|alphaVel|}} \\
 fratelimit_4 &= \min(C, D)
 \end{aligned}$$

The *fratelimit_4* is dependent on both dynamical and geometrical constraints. The dynamical constraints are in the maximum allowable axial accelerations of the machine as covered by $xAcc_{max}$ and $yAcc_{max}$. The geometrical constraints are in $alphaVel(u)$, $betaVel(u)$ and $\rho(u)$.

3.20 Feedrate rising S-curve

In CNC machining operation, the feedrate does not rise abruptly from zero. A smooth feedrate rise is required to ensure machine stability. A sigmoid S-shaped curve was selected for the rise of the current running feedrate to the current feedrate limit.

$$curr_frate(rsu) = \frac{curr_frate_limit(rsu)}{\left(1 + e^{(-rsu * rshape1)}\right)^{rshape2}}$$

where the applicable variable range is

$$rsu_start_rise \leq rsu \leq rsu_end_rise$$

The linear parameter transformation from u to rsu is as follows:

$$rsu = rm * u + rkconst$$

To ensure smoothness of the rising feedrate curves, simulations were conducted to determine the best values of the parameters below. These variables are user specified.

Table 3.17: Rising S-curve parameters

<code>rsu_start_rise</code>	$= 0.00$	beginning of rising S-curve
<code>rsu_end_rise</code>	$= 0.05$	end of rising S-curve
<code>rshape1</code>	$= 5.00$	S-curve smoothness shaping factor
<code>rshape2</code>	$= 8.00$	S-curve smoothness shaping factor
<code>rsu1</code>	$= 0.00$	start of u linear transformation
<code>rsu2</code>	$= 3.00$	end of u linear transformation
<code>rm</code>	$= \text{calc}$	slope calculated for each parametric curve
<code>rkconst</code>	$= \text{calc}$	constant calculated for each parametric curve
<code>rsu</code>	$= \text{calc}$	transformed variable in the rising S-curve

3.21 Flowchart of main interpolation program

The main program flowchart for the parametric curve interpolation is shown in Fig [3.24]. The parameter update $u = u + u_{\text{next}}$ is executed for each loop. As shown in the main flowchart, there are 3 main processing modules that cover computations involving u , that is, in the rising, main and falling feedrate sections.

The flowchart for the feedrate rising section is shown in Fig [3.25], and the feedrate falling section in Fig [3.33]. The computations for the main section are further separated into 3 cases described below.

1. When the current_feedrate is above feedrate_limit, called CaseA_(Above), the flowchart is Fig [3.26] and is continued in Fig [3.27].
2. When the current_feedrate is below feedrate_limit, called CaseB_(Below), the flowchart is Fig [3.28] and is continued in Fig [3.29].
3. When the current_feedrate is equal to the feedrate_limit, called CaseE (Equal), the flowchart is Fig [3.30].

The processing in CaseA_(Above) is basically to push down the running current_feedrate to just below the calculated feedrate_limit. On the other hand, the processing in CaseB_(Below) is to push up the current_feedrate to also be just below the calculated feedrate_limit. The net result is the current_feedrate will be very close but just below the calculated feedrate_limit. This is the feedrate constraint part of this work.

A common module named Flowchart_Calculate_u_next(u), is shown in Fig [3.31]. This module will be invoked by the feedrate rising, main and falling modules. This is the implementation of Call-and-Return (CAR) software architecture.

The Flowchart_Calculate_u_next(u) module will invoke another module named Flowchart_Calculate_u_next(u)_with_eps_pushdown, as shown in Fig [3.32]. This last module is responsible for pushing down chord-error (epsilon) below the error tolerance of $(10)^{-6}$. This is the chord-error constraint part of this work.

Note that a successful effort was also made to push up the value of chord-error to very close but just below the chord-error tolerance, similar to the execution for feedrate constraint. However, this is not a wise move, since increasing the chord error means increasing both total chord-error and total chord length. An increase in chord length means a decrease in total number of interpolated points, a thing that is favorable.

However, results showed that there were jitters (severe fluctuations) in feedrate at several segments along the path. The feedrate profile was no longer smooth. Knowing that the chord-error is already below tolerance, the advice taken is to ignore the push up (keep it if it is already below tolerance). And it is sensible because this would keep the total chord-error small and the feedrate profile smooth. Therefore, the chord-error push up idea was abandoned. Only a chord-error push down was executed in the algorithm.

It must be the "natural" computation effect in the second order Taylor's expansion for u_{next} that generates chord-errors far below error tolerance. Thus, there is no need for intervention to push it up. In addition, this effect ensures smoothness of the feedrate profiles of all curves in this work.

This section discusses all the flowcharts in the interpolation algorithm beginning from Fig [3.24] until Fig [3.32]. Notice that the algorithm is modular and developed in a structured manner following good software engineering practices. The decision points are well structured and not haphazard, which can cause serious problems like side effects. The problem of side effects will be explained in the section Appendix [App.1.1].

An interesting note in Fig [3.24], the main program flowchart for the entire interpolation, is the existence of "EXIT ON LOGIC ERROR". This exit is an impossible exit, because logically and mathematically it cannot happen. But it was implemented anyway to catch any computational quirks since the algorithm is supposed to cater for ten(10) different parametric curves of various complexities. This quirk can be caused by computations involving numbers below machine-epsilon value. On addition and subtraction, these very small numbers, are treated by computers as "zero", even though the numbers are just very small but truly non-zero. Please refer to Section[3.26] on Software engineering practice that discusses machine-epsilon effects.

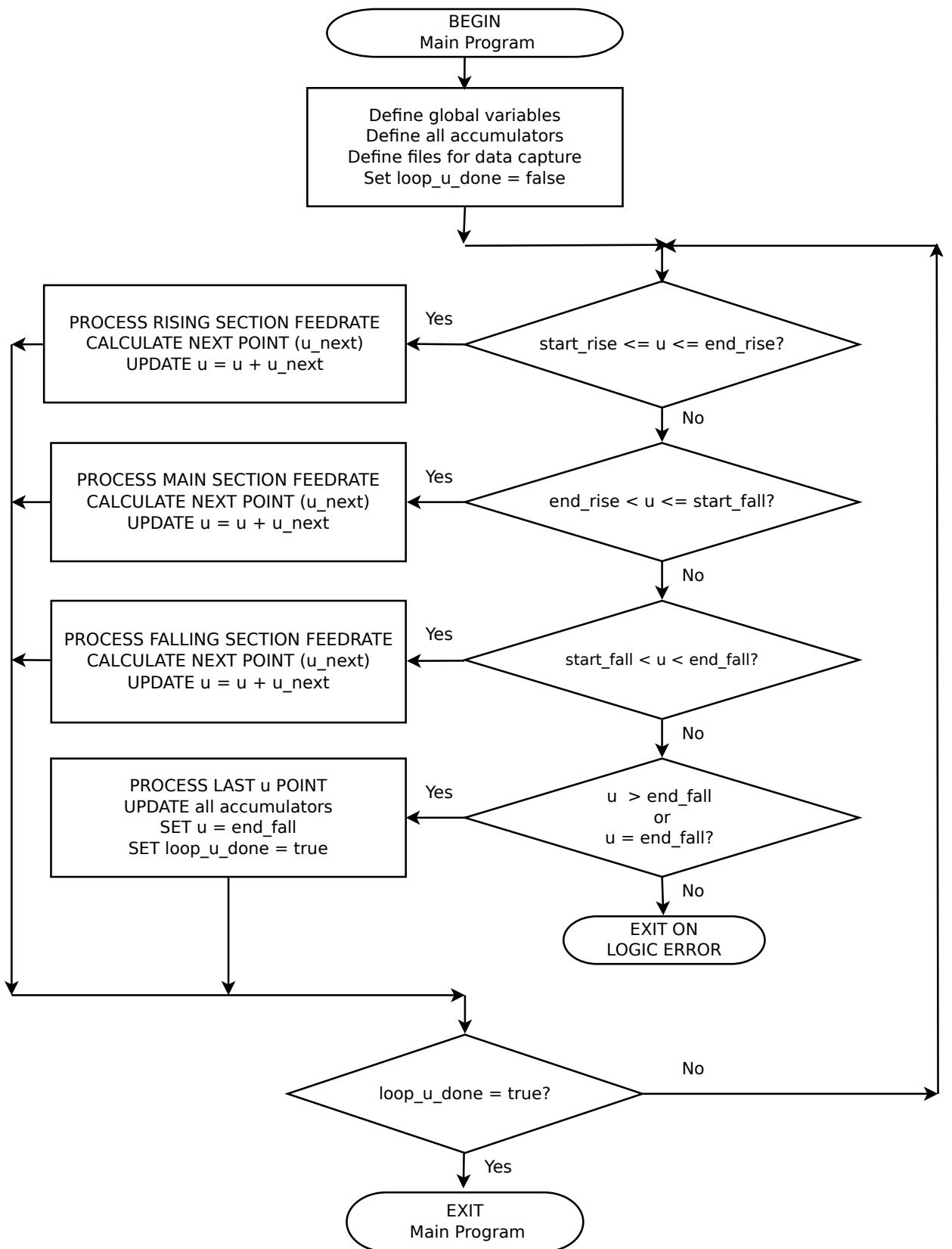
The impossibility of this "EXIT ON LOGIC ERROR" can be explained as follows. Consider a real number "M", that is compared to another real number "K".

Logically, only three(3) possibilities can happen:

- (1) M equals K
- (2) M greater than K
- (3) M less than K

In the case of, "EXIT ON LOGIC ERROR", M is none of the above three(3) possibilities. Thus, it is logically and mathematically impossible. Even though considered impossible, this exit was incorporated to handle quirks and analyze their corresponding causes.

Figure 3.24: Flowchart of main program



3.22 Feedrate Limit algorithm regions

The flow chart for the main program shows that the range of processing is divided into 3 separate regions for $0.0 \leq u \leq 1.0$ as follows.

1. Region ($start_rise \leq u \leq end_rise$), the feedrate rising S-curve region.
2. Region ($end_rise < u \leq start_fall$), the feedrate dynamic variation main region.
3. Region ($start_fall < u \leq end_fall$), the feedrate falling S-curve region.

where $start_rise = 0.0$ and $end_fall = 1.0$ with (end_rise and $start_fall$) user selected.

For all parametric curves considered, a large majority of computations for the interpolation algorithm occurs in the feedrate dynamic variation or main region. The feedrate profiles vary markedly from curve to curve.

The rising S-curve and falling S-curve sections are regions where the *feedrate_limit* are determined by the user assigned rising and falling sigmoid or S-curves. The *feedrate_limit* is assigned the exact value according to the rising or falling S-curves. The current or running feedrates are still computed to ensure that the chord-error and running feedrate constraints are observed. In this work, 4 different values of FCs were considered (FC = 10, 20, 30, 40). The execution results are provided in Chapter 4.

It is important to note that CaseA (Above) and CaseB (Below) are the conditions before processing. During processing, all *current_feedrates* are either "iteratively pushed-down" to below and very close to its *feedrate_limit*, or "iteratively pushed-up" to below and very close to its *feedrate_limit*.

Thus, it effectively means without exception, all *current_feedrates* running in operation are below the *feedrate_limit* throughout the traversal of the parametric curve. The execution results will be shown in Chapter 4. This condition is called "feedrate constrained", and is one of the main objectives of this thesis.

Figure 3.25: Flowchart of Feedrate Rising S-Curve

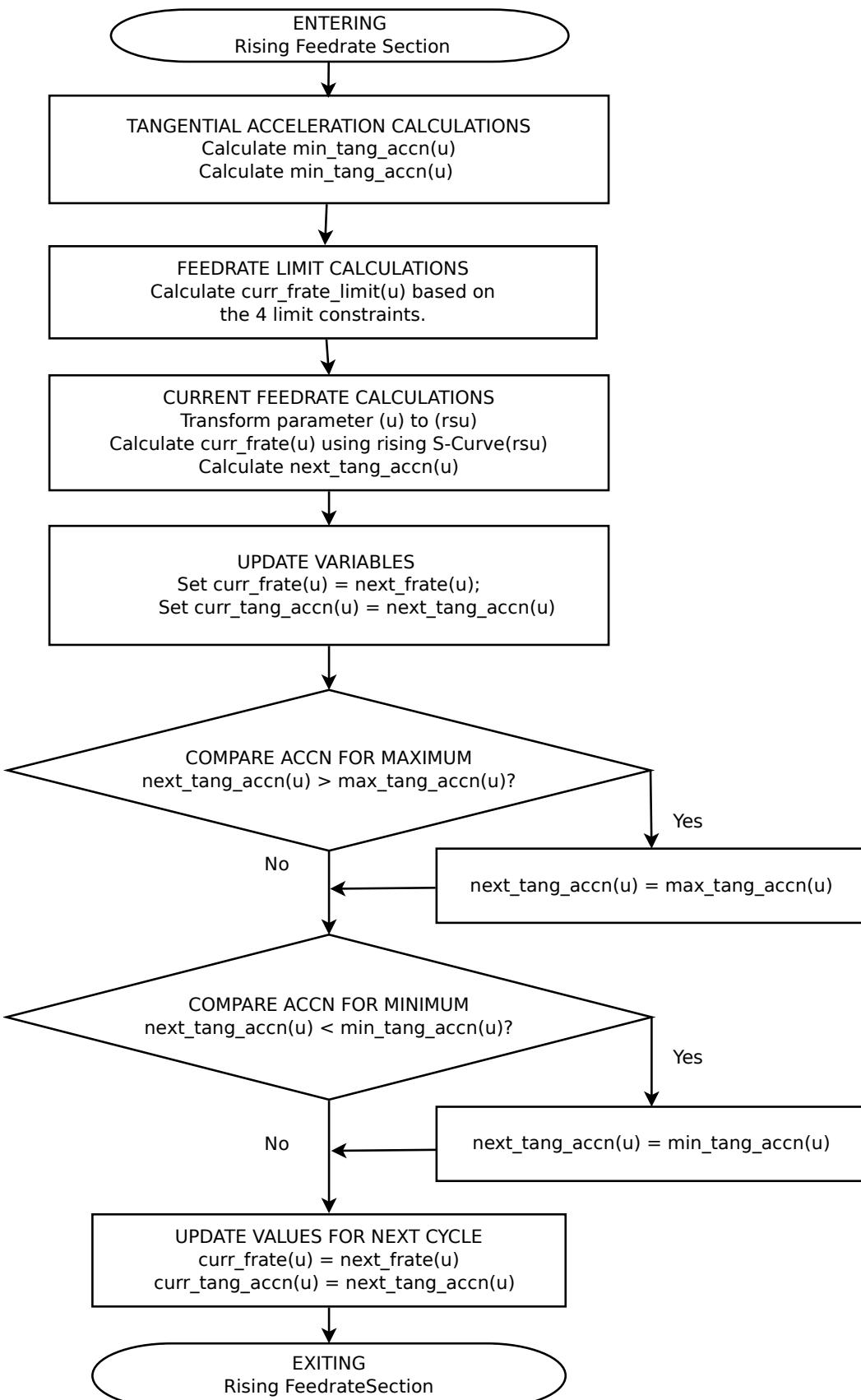


Figure 3.26: Flowchart Current Feedrate above Feedrate Limit Part 1/2

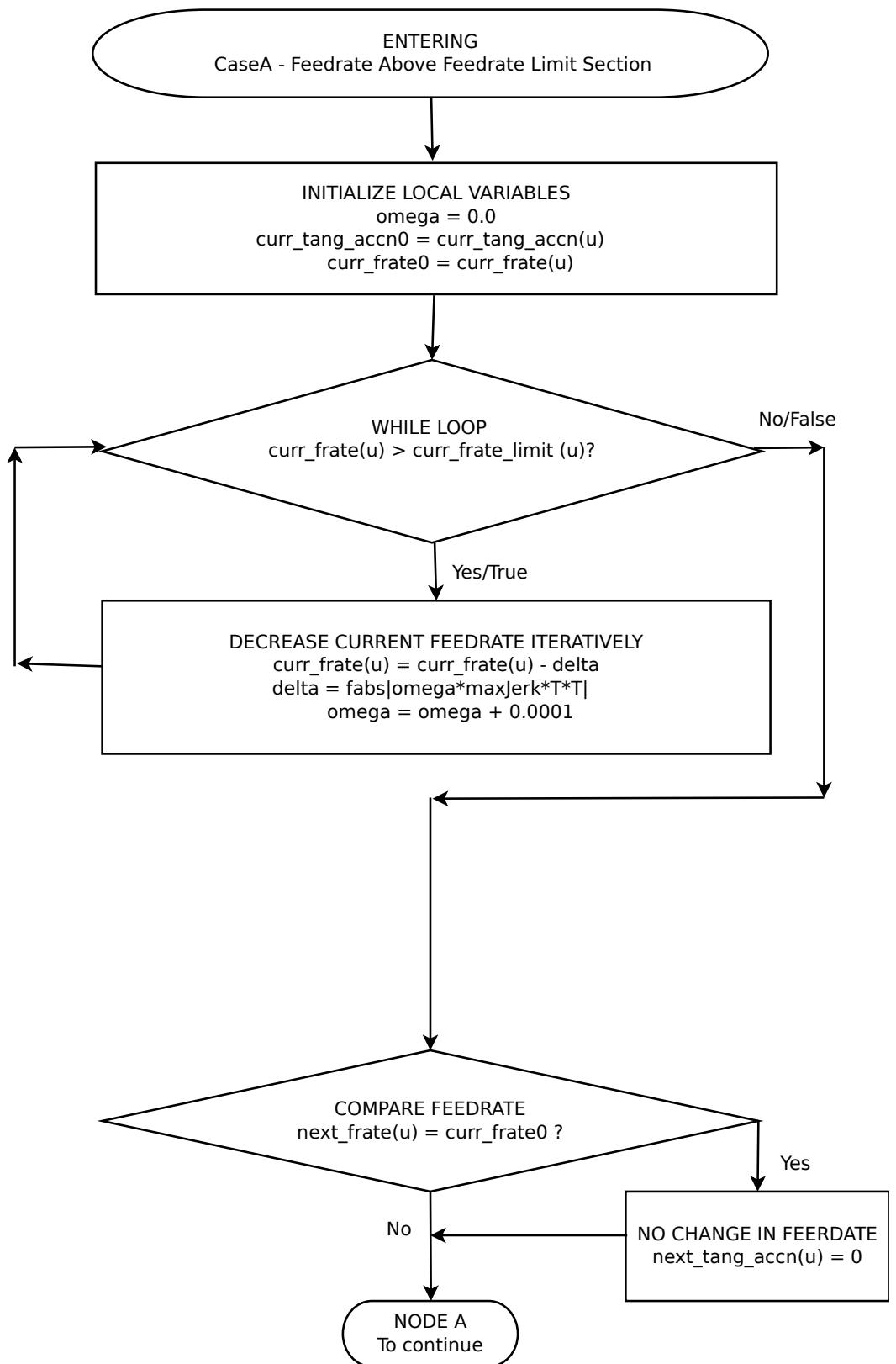


Figure 3.27: Flowchart Current Feedrate above Feedrate Limit Part 2/2

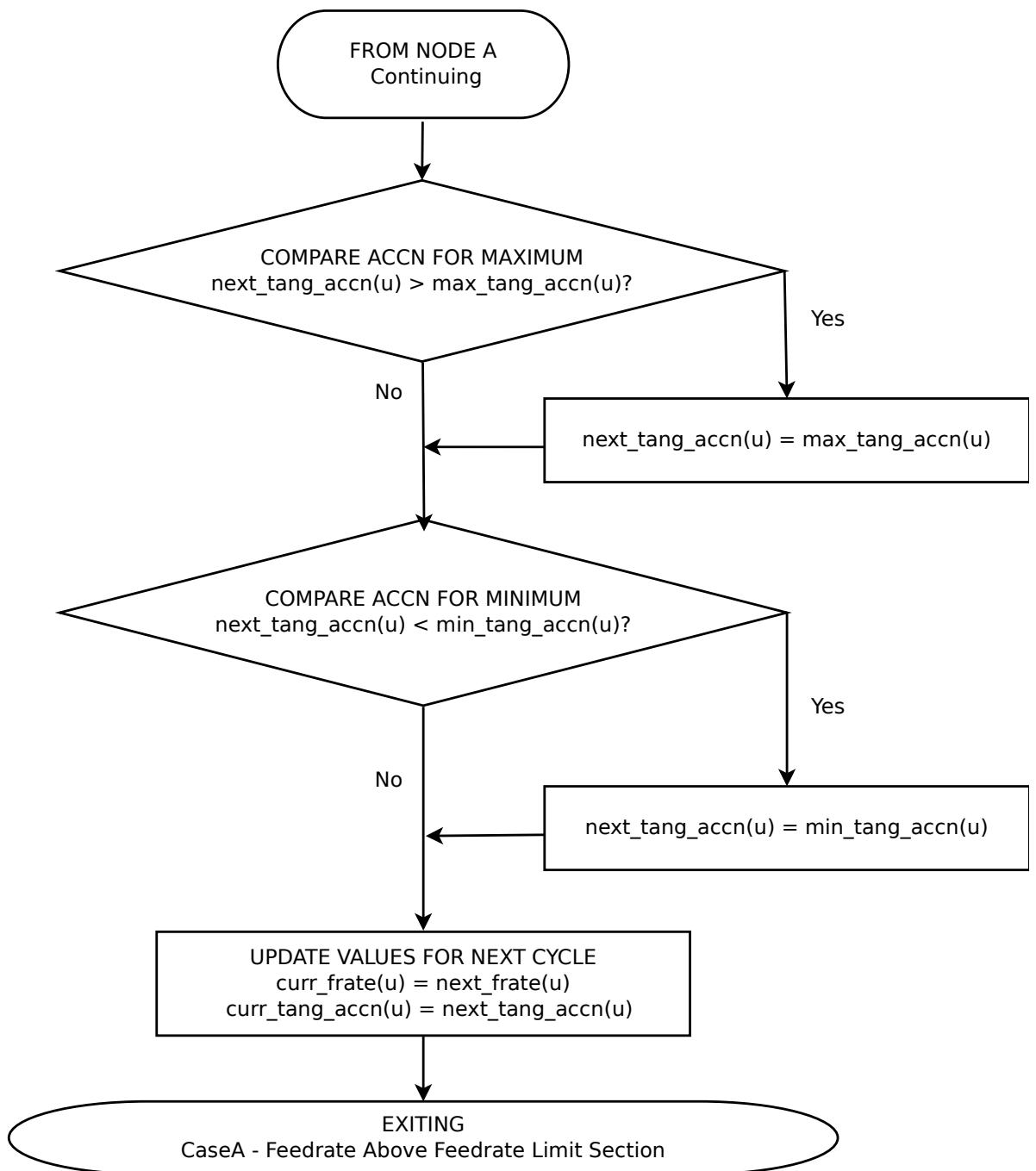


Figure 3.28: Flowchart Current Feedrate below Feedrate Limit Part 1/2

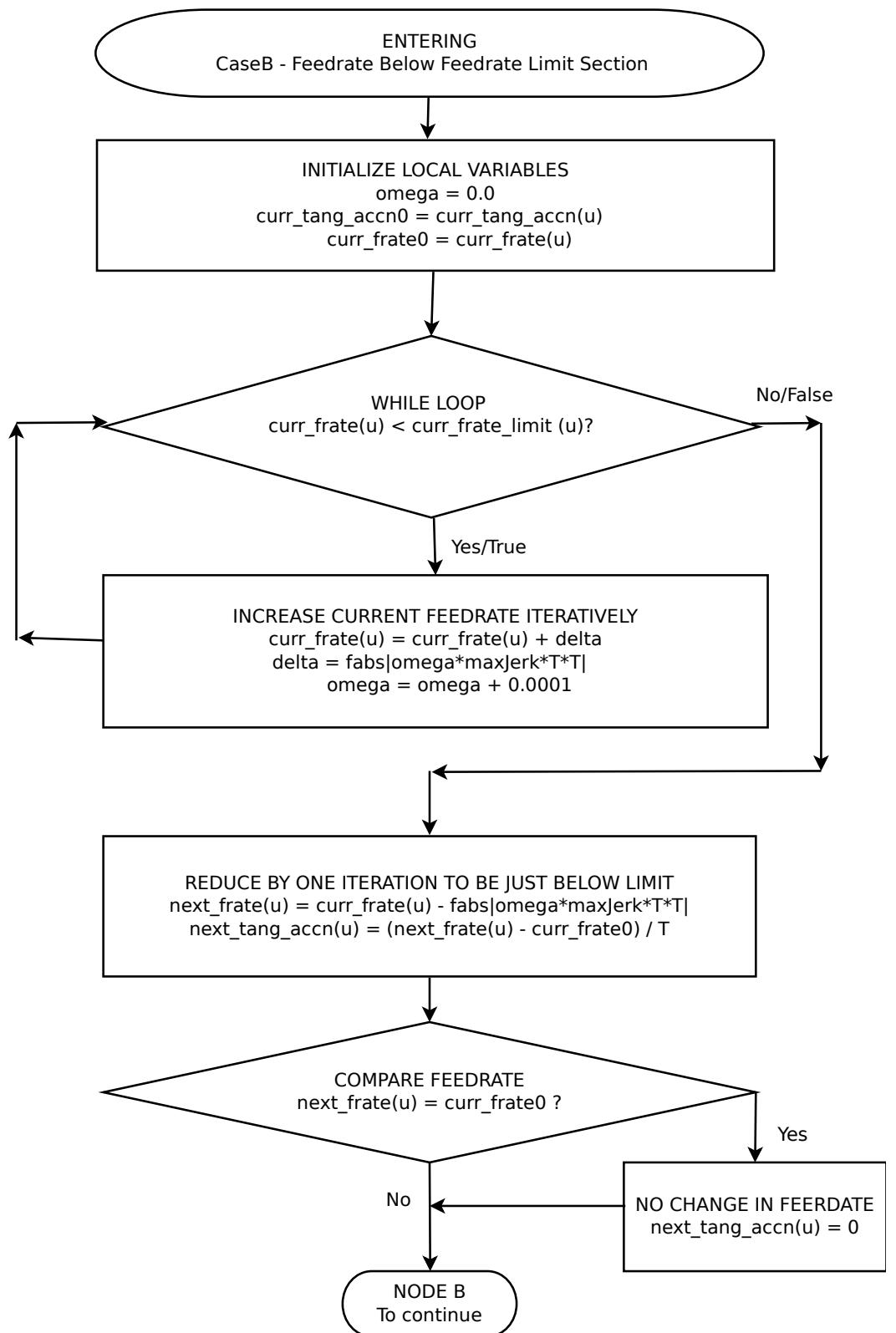


Figure 3.29: Flowchart Current Feedrate below Feedrate Limit Part 2/2

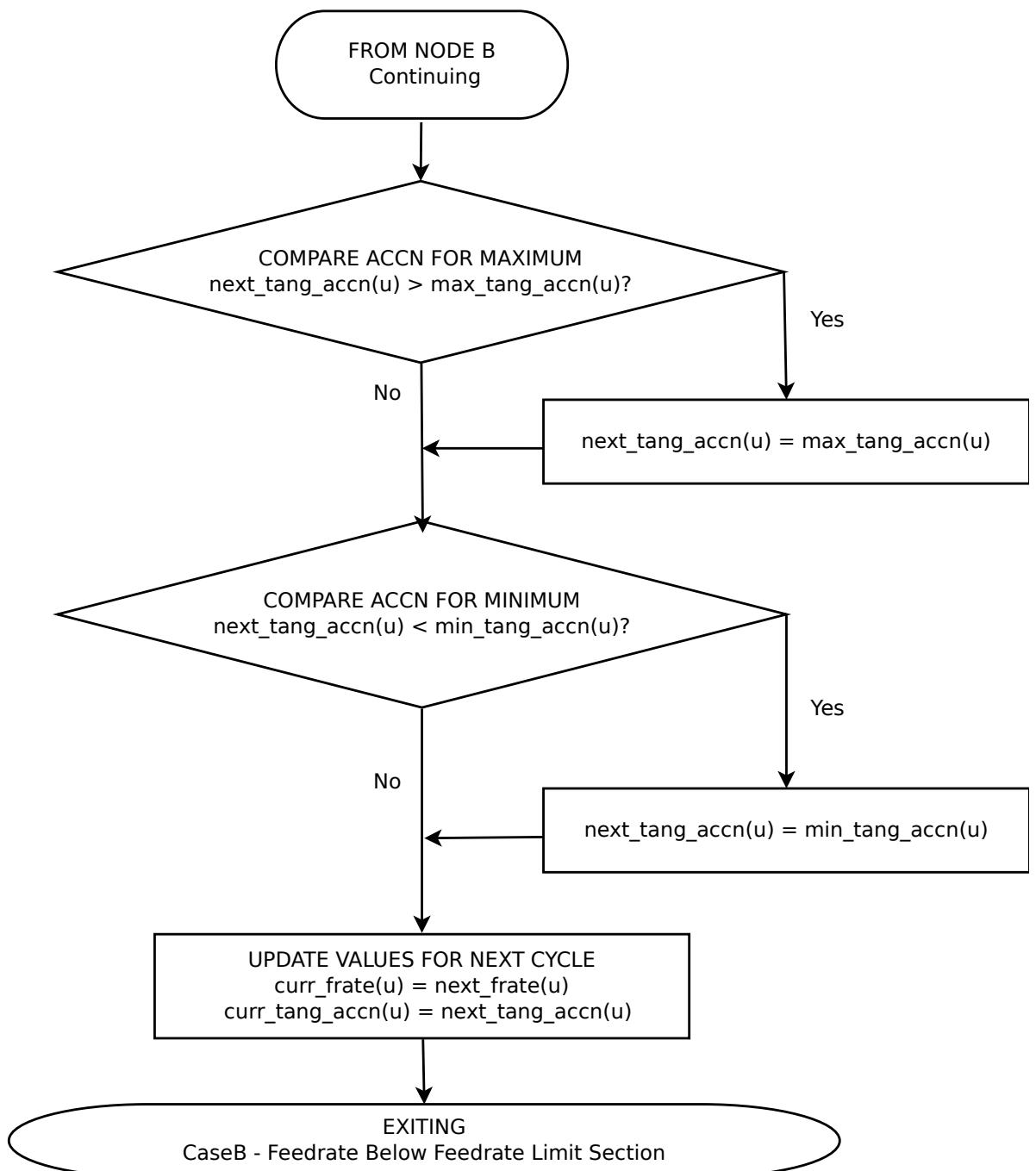


Figure 3.30: Flowchart Current Feedrate equal Feedrate Limit

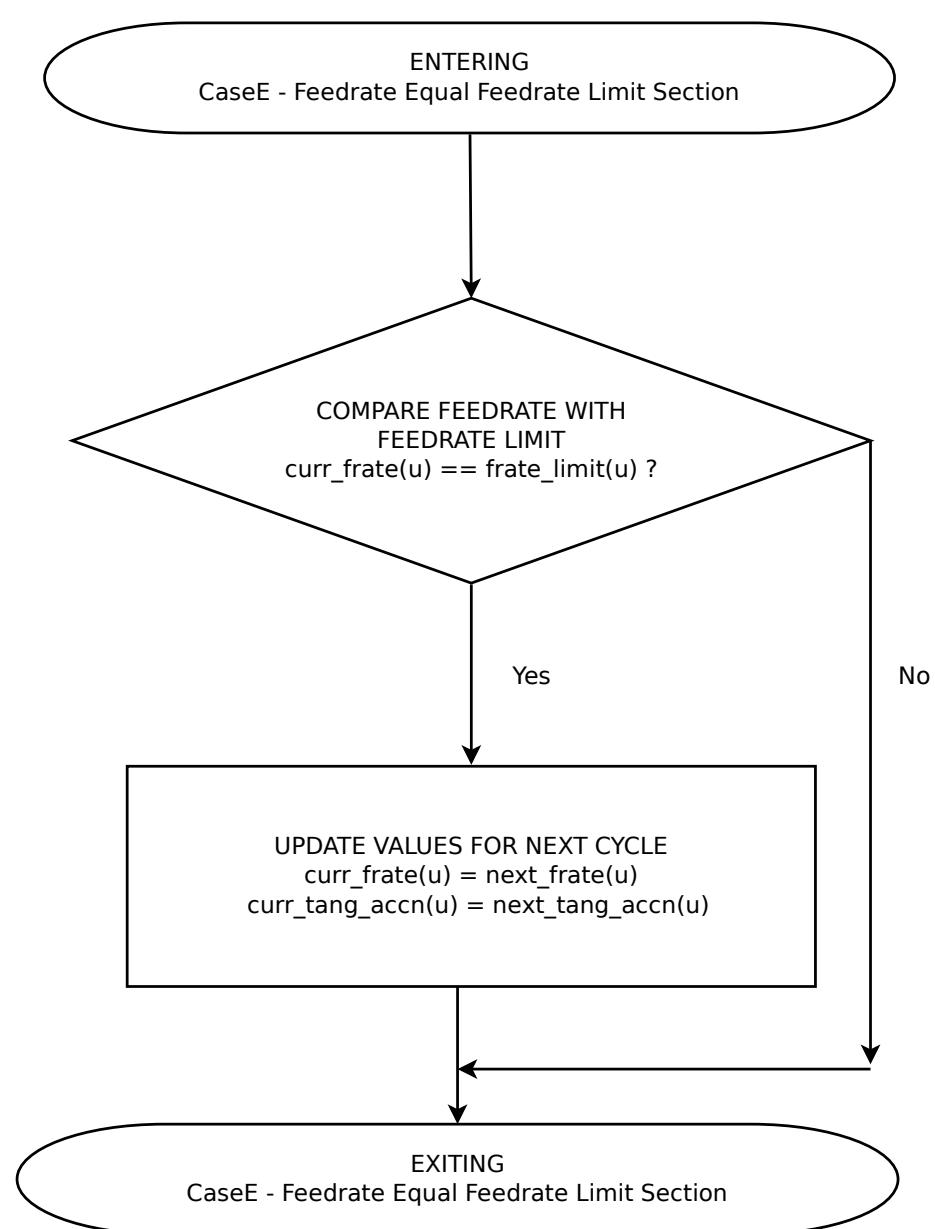


Figure 3.31: Flowchart Calculate u_next(u)

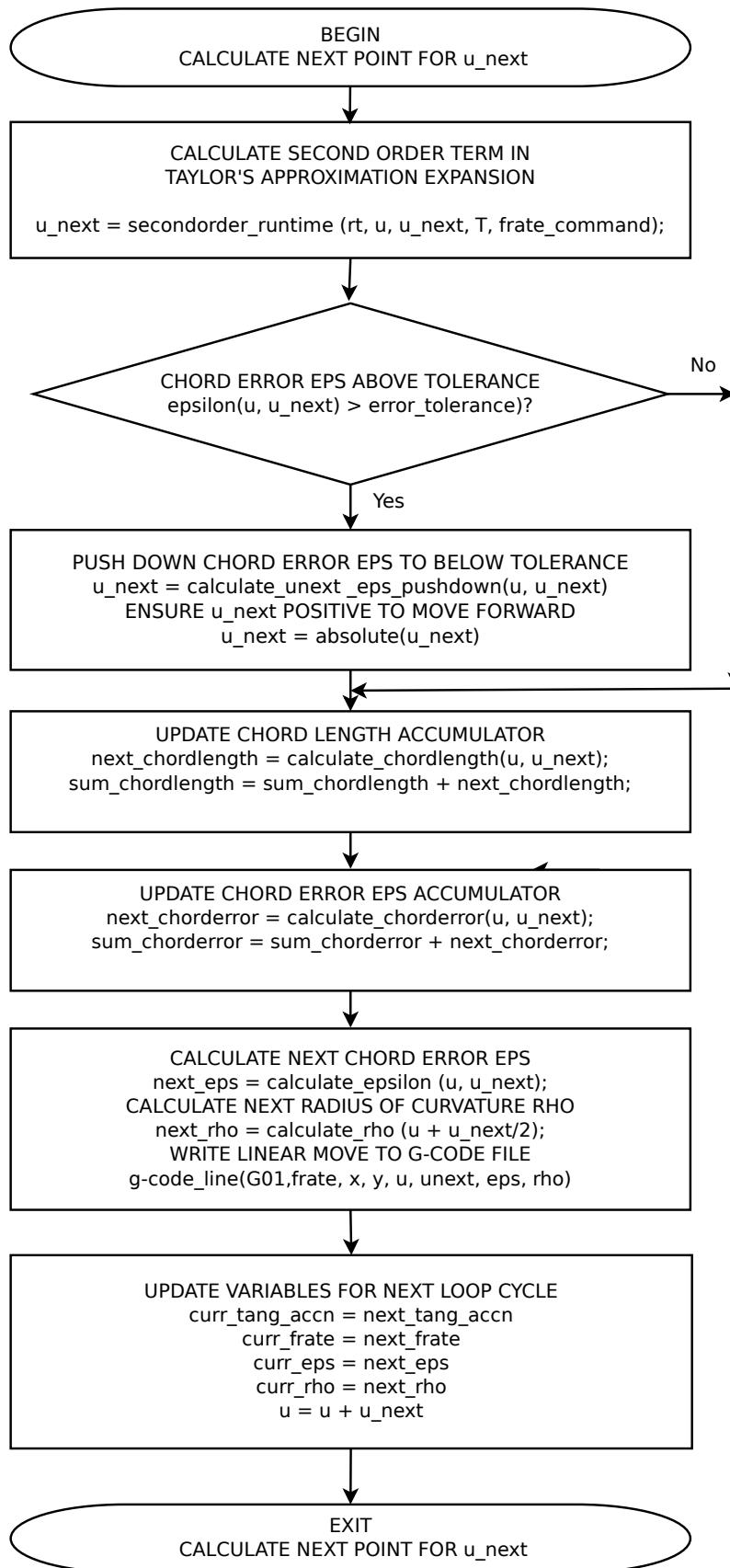


Figure 3.32: Flowchart Calculate $u_{\text{next}}(u)$ with ϵ s pushdown

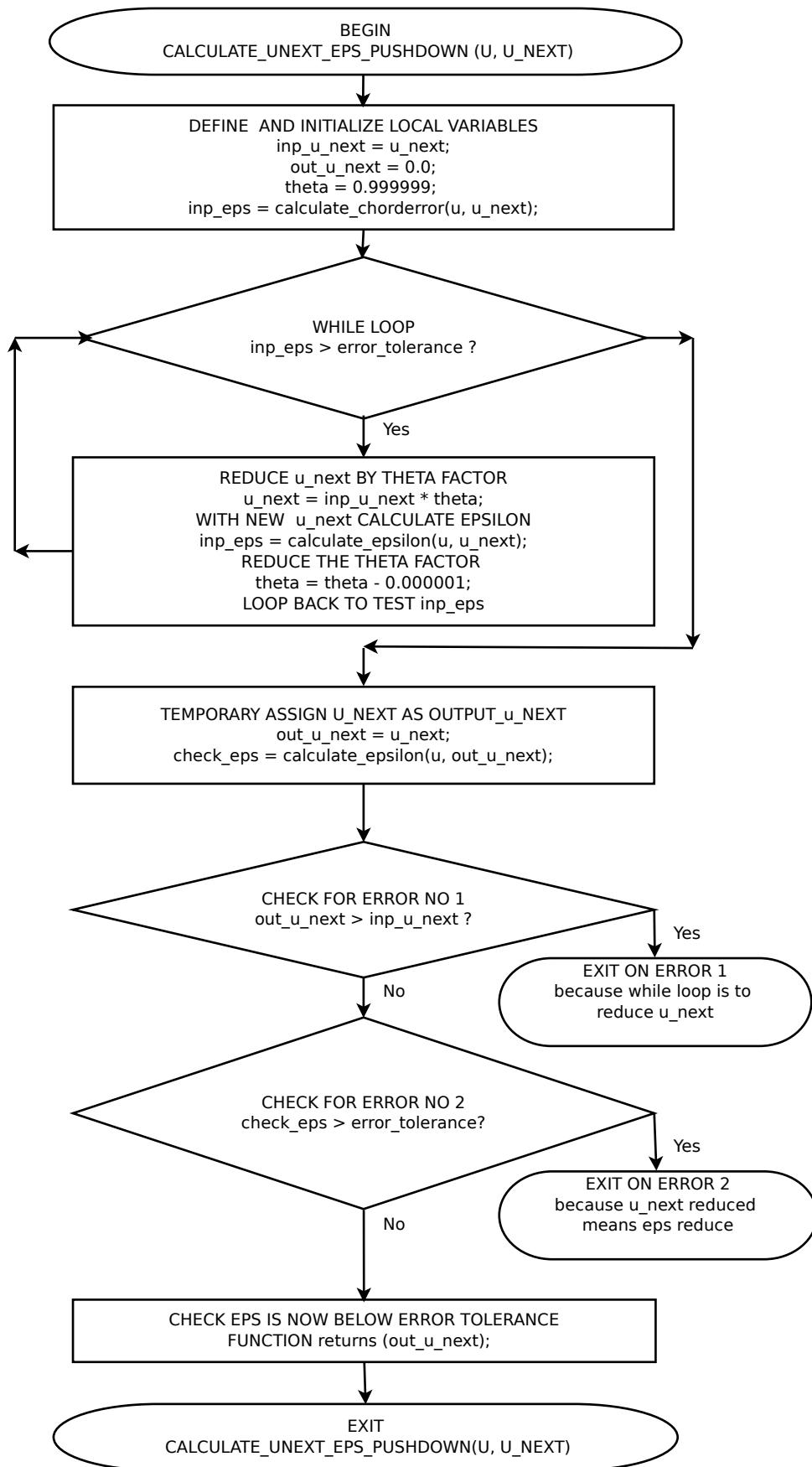
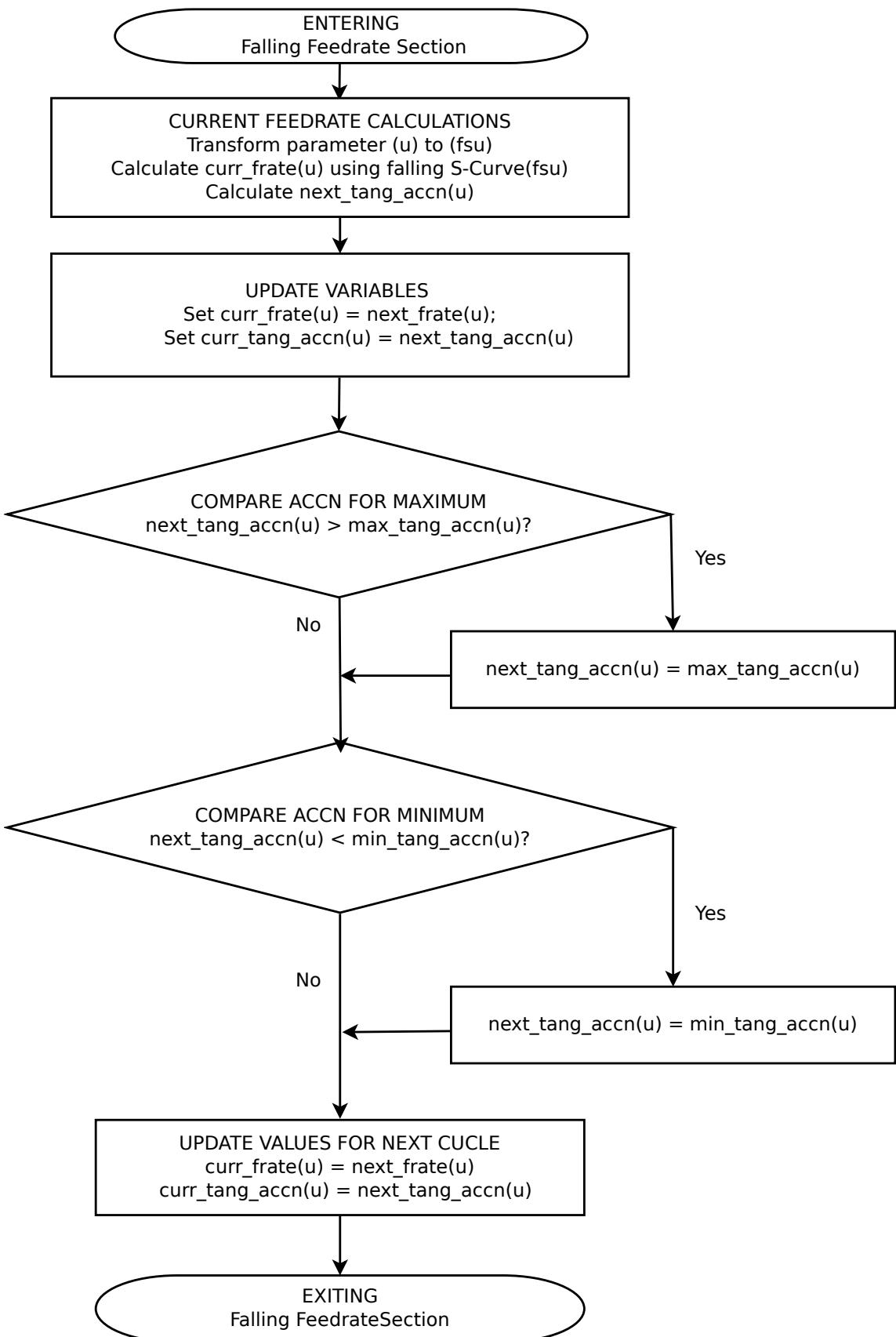


Figure 3.33: Flowchart of Feedrate Falling S-Curve



3.23 Feedrate falling S-curve

The feedrate must not fall abruptly to zero. A smooth feedrate fall is required to ensure machine stability. The S-shaped curve below was selected for the fall of the current running feedrate to zero.

$$curr_frate(fs_u) = 1 - \frac{curr_frate_limit(fs_u)}{\left(1 + e^{(-fs_u * fshape1)}\right)^{fshape2}}$$

where the applicable variable range is

$$fs_u_start_fall \leq fs_u \leq fs_u_end_fall$$

The linear parameter transformation from u to fs_u is as follows:

$$fs_u = fm * u + fkonst$$

To ensure smoothness of the falling feedrate curves, simulations were conducted to determine the best values of the parameters. These variables are user specified.

Table 3.18: Falling S-curve parameters

<code>fsu_start_fall</code>	$= 0.95$	beginning of falling S-curve
<code>fsu_end_fall</code>	$= 1.00$	end of falling S-curve
<code>fshape1</code>	$= 5.00$	S-curve smoothness shaping factor
<code>fshape2</code>	$= 8.00$	S-curve smoothness shaping factor
<code>fsu1</code>	$= 0.00$	start of u linear transformation
<code>fsu2</code>	$= 3.00$	end of u linear transformation
<code>fm</code>	$= \text{calc}$	slope calculated for each parametric curve
<code>fkonst</code>	$= \text{calc}$	constant calculated for each parametric curve
<code>fsu</code>	$= \text{calc}$	transformed variable in the rising S-curve

3.24 Acceleration constraint lambda safety factor

Table 3.19: Acceleration safety factor (lambda)

$$\begin{aligned}
 \textit{lamda} &= \textit{user_select_safety_factor}(0.0\text{to}1.0) \\
 C &= \sqrt{\frac{\textit{lamda} * \textit{rho} * \textit{xAcc}_{max}}{|\textit{betaVel}|}} \\
 D &= \sqrt{\frac{\textit{lamda} * \textit{rho} * \textit{yAcc}_{max}}{|\textit{alphaVel}|}} \\
 \textit{fratelimit_4} &= \textit{minimum}(C, D)
 \end{aligned}$$

The *fratelimit_4* is about acceleration confinement within a (min, max) range of permissible accelerations. It is dependent on the combined dynamical and geometrical constraints. The dynamical machine constraints are in the maximum allowable axial accelerations of the machine as covered by $xAcc_{max}$ and $yAcc_{max}$. The geometrical constraints are in $alphaVel(u)$, $betaVel(u)$ and $rho(u)$, since u changes along the curve these geometrical values also change.

The choice for the value of lambda, the safety factor for containment of acceleration, makes a big impact on the occurrence of jerks (jitters) in the feedrate. The values of 0.10, 0.18, 0.20 and 0.50 for lambda were tested for the algorithm runs to eliminate the jerks. A lower value of lambda will eliminate jerks but will reduce the feedrate significantly. The results for the threshold value for lamda is provided in Chapter 4 at reference link [4.3.4].

3.25 Four(4) Algorithm Performance Metrics

There are four(4) metrics to assess the overall algorithm performance in this work. The first three terms are ratios while last term is a percentage measure of the difference.

1. SCE/TIP : This is the ratio of total sum-chord-error divided by the total number of interpolated points.
2. SCE/SCL : This is the ratio of total sum-chord-error divided by the total sum-chord-length.
3. SAA/SCL : This is the ratio of the sum-arc-areas divided by the total sum-chord-length.
4. 100(SAL-SCL)/SAL : This is the ratio of the difference between the sum-arc-length and the sum-chord-length, divided by the sum-arc length and multiplied by 100 to represent it in percentage form.

The calculation for the individual arc-length (AL) is provided in section [3.25.1]. The calculation for the individual chord-length (CL) is provided in section [3.25.2]. The calculation for the individual arc-theta (AT) is provided in section [3.25.3]. The calculation for the individual arc-area (AA) is provided in section [3.25.4].

Consider these two(2) important terms. The term NAL(u) is the next-arc-length or length of the arc calculated by the algorithm from point u to the point u + (u next). Similarly, NCL(u) is the next-chord-length or length of the chord calculated by the algorithm from point u to the point u + (u next).

By definition, the sum-arc-length SAL(u), is the cumulative sum of NAL(u) from NAL(0.00) to NAL(u), where u is incremented by (u-next) in each step. Similarly, the sum-chord-length SCL(u), is the cumulative sum of NCL(u) from NCL(0.00) to NCL(u). Generally, we have the following definition of terms in this work.

Table 3.20: Definitions of SAL(u), SCL(u), SAT(u), SAA(u) and SCE(u)

$$\begin{aligned}
 SAL(u) &= \sum_{k=0.00}^u NAL(k) \\
 SCL(u) &= \sum_{k=0.00}^u NCL(k) \\
 SAT(u) &= \sum_{k=0.00}^u NAT(k) \\
 SAA(u) &= \sum_{k=0.00}^u NAA(k) \\
 SCE(u) &= \sum_{k=0.00}^u NCE(k)
 \end{aligned}$$

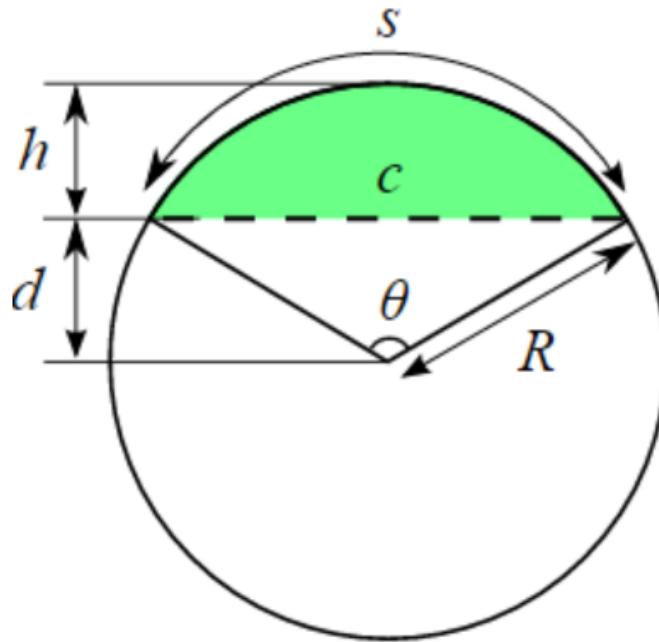
Table 3.21: Definitions of Total SAL, Total SCL, Total SAT, Total SAA, and Total SCE

$$\begin{aligned}
 Total_SAL &= \sum_{k=0.00}^{k=1.00} NAL(k) \\
 Total_SCL &= \sum_{k=0.00}^{k=1.00} NCL(k) \\
 Total_SAT &= \sum_{k=0.00}^{k=1.00} NAT(k) \\
 Total_SAA &= \sum_{k=0.00}^{k=1.00} NAA(k) \\
 Total_SCE &= \sum_{k=0.00}^{k=1.00} NCE(k)
 \end{aligned}$$

Note that the variable k is discrete and non-uniform. In fact, the successive values of k are exactly the successive values of u, generated by the interpolation algorithm, that is, while simultaneously constraining chord-error and running feedrate.

Saying it is a straightforward manner, the values of k are exactly the values of the interpolated points u. That is the reason for k being non-uniformly spaced. Finding this k points is in fact, the main objective of the work in this thesis.

Figure 3.34: Calculation for the Circle Arc-Length



3.25.1 Arc-Length Approximate calculation

The first-order approximation for the calculation of the arc-length $AL(u, u_{next})$ segment of a generic curve with equation $C(x(u), y(u))$, where $x(u)$ and $y(u)$ are known is given by:

$$AL(u, u_{next}) = (u_{next}) * \sqrt{\left(\frac{dx}{du}\right)^2 + \left(\frac{dy}{du}\right)^2}$$

Listing 3.1: Implementation for Approximate Arc-Length Calculation

```
// =====
double fxn_calc_arc_length(double u, double u_next)
// =====
{
    // LOCAL VARIABLES
    double arc_length = 0.0;

    // GEOMETRIC CALCULATION ARC LENGTH OF PARAMETRIC CURVES
    double dx_du = fxn_cvel_x(u);
    double dy_du = fxn_cvel_y(u);
    double sumsquare = (dx_du)*(dx_du) + (dy_du)*(dy_du);
    double arc_length0 = (u_next)*sqrt(sumsquare);
    arc_length = fabs(arc_length0);

    return (arc_length);
}
```

3.25.2 Chord-Length Exact calculation

The exact calculation for the chord-length $CL(u, u_{next})$ of a generic curve with equation $C(x(u), y(u))$, between two points on the curve at (x_1, y_1) and (x_2, y_2) is given by:

$$CL(u, u_{next}) = \sqrt{\left(x_2 - x_1 \right)^2 + \left(y_2 - y_1 \right)^2}$$

Listing 3.2: Implementation of Exact Chord-length Calculation

```
// =====
double fxn_calc_chordlength_use_paramcurve (double u, double u_next)
// =====
{
    double chordlength;
    double x1, x2, y1, y2;

    // Use param curve and geometry
    x1 = fxn_cpos_x (u);
    y1 = fxn_cpos_y (u);
    x2 = fxn_cpos_x(u+u_next);
    y2 = fxn_cpos_y(u+u_next);

    // Pythagoras theorem for a right angled triangle
    chordlength = fabs(sqrt(pow((x2-x1), 2.0) + pow((y2-y1), 2.0)));

    return (chordlength);
}
```

3.25.3 Arc-Theta Approximate calculation

The equation for the approximate arc-theta requires chord-length CL, and the radius of curvature rho values. The equation used in this calculation is based on website reference [circular_segment_equation_and_calculator].

$$Arc_Theta(CL, R) = (2) * \sin^{-1}(CL/2R)$$

where R = approximately Radius of Curvature rho, and the chord-length CL is calculated from a previous equation.

Listing 3.3: Implementation of Approximate Arc-Theta Calculation

```
// =====
double fxn_calc_arc_theta(double u, double chord_length, double rho)
// =====
{
    double arc_theta = 0.0;
    double arg_arcsine = chord_length/(2.0*rho);

    if ((arg_arcsine < -1.0) || (arg_arcsine > +1.0) )
    {
        printf("ERROR: Angle_theta is out of range [-1.0 : +1.0] \n");
        printf("Value of angle arg_arcsine = %.12e \n", arg_arcsine );
        exit (1);
    } else {
        arc_theta = 2.0*asin(arg_arcsine );
    }

    return( arc_theta );
}
```

3.25.4 Arc-Area Approximate calculation

The arc-area is the area bounded by the chord and the arc-segment. The equation for the approximate arc-area requires the radius of curvature rho and the angle theta in radians subtended by the arc segment. The equation used in this calculation is based on website reference [circular_segment_equation_and_calculator].

$$Arc_Area(R, \theta) = (R^2/2) * (\theta - \sin(\theta))$$

where R = approximately Radius of Curvature rho, and the subtended angle theta is in radians.

Listing 3.4: Implementation of Approximate Arc-Area Calculation

```
// =====
double fxn_calc_arc_area(double u, double chord_length, double rho)
// =====
{
    double arc_area = 0.0;

    // GET VALUE OF ARC-THETA FROM PREVIOUS FUNCTION
    double arc_theta = fxn_calc_arc_theta(u, chord_length, rho);
    double DIFF = arc_theta - sin (arc_theta);

    // CALC ARC AREA
    arc_area = (rho*rho) * (DIFF)/2;

    return (arc_area);
}
```

3.26 Software engineering practice

The premise for the conduct of this thesis is to have a software design that is structured, fully functionalized and modularized. The use of GNAT Studio 2021 as the Integrated Development Environment (IDE) allows a combination of C, C++ and Ada code to be compiled together into a single executable. Ada is known for safe and secure codes and includes its own realtime library.

In general, software code implementations must not only run correctly, but must also run efficiently. Therefore, design of the codes must follow good software engineering practices. This thesis is under the category of mechatronics and systems design, so correct software codes is an important and critical part of the work.

The algorithm design must be highly structured, functionalized and modularized. This practice makes program execution flow readable and understandable. The resulting flowcharts will provide easy error tracing, adaptations, and additions of new functionality.

For example, inexperienced software programmers forget the simple rule that every time a function returns from a call, then all values of local variables are wiped out from memory. This requires that local variables be saved globally before the called function returns. This is a basic computing principle.

Technically, when a function executes, it may add some of its local state data to the top of the stack; when the function exits it is responsible for removing that data from the stack. In other words, since the stack memory of a function gets deallocated after the function returns, there is no guarantee that the value stored in those area will stay the same. References: A. A. Wikipedia (2023) and J. Chen and Guo (2023).

Another issue is the concept of zero in computing machines. In codes using floating-point representation of real numbers, any number below machine-epsilon (macheps), is treated as zero by the machine for addition and subtraction operations. Machine-epsilon is actually a very small but non-zero number, that varies from machine to machine. In the work conducted in this document at reference [App.2.2], the machine-epsilon (macheps) for 64-bit double type was found to be $2.22(10)^{-16}$. This means an algorithm check for addition or subtraction by zero is actually a check of addition or subtraction of very small non-zero numbers with values below machine-epsilon. Reference: B. A. Wikipedia (2023).

The side-effect phenomena in computing is another important issue. In computer science, an operation, function or expression is said to have a side effect if it modifies some state variable value(s) outside its local environment. This means, there are additional effects other than its primary effect of just returning a value to the caller of the operation. Because the algorithm in this work uses many global and local variables, it can be difficult to track changes and side-effects, that is, if the algorithm is not designed properly. Reference: C. A. Wikipedia (2023).

A decision operation (diamond flowchart symbol) must only have two outputs, True or False. It is a mistake, in fact a serious error not to have two branches as outputs.

The algorithm must be designed to exit when there is an error at any point in its execution because all subsequent (downstream) calculations can be considered erroneous (useless) due to this upstream error.

Inclusion of error trapping functions is vital in any processing code. The algorithm in this work, for example, traps errors when u_{next} repeatedly stays at "zero" in five(5) consecutive loops in Taylor's expansion calculation. This means the interpolation step does not move forward to the next point. This is caused by the effect of machine epsilon. The "zero" here does not mean real zero, but is the computer machine-epsilon, the smallest non-zero value the computer can represent. Any value below this machine-epsilon is treated as zero for additions and subtractions by the computer.

Algorithm verification through accounting is the term used in this work to check that all calculations tally to the correct expected total value. As an example, the algorithm was built to check that the histogram total sum value (of interpolated points) tallies with the total number of interpolated points. Similarly, the histogram total sum value (of points above feedrate limit, and points below feedrate limit) tallies with the total number of interpolated points. In the same manner, the histogram total sum value (of points where chord-errors are all below tolerance) tallies with the total number of interpolated points.

In this work, the algorithm verification conducted above proves the achievement in simultaneously constraining chord-error and feedrates, thus meeting the objective of the interpolation algorithm.

3.27 Design of Algorithm Codes

3.27.1 Algorithm Text Reports

Every combination of the set (curve type, feedrate command FC and lamda safety factor) made as input to the algorithm will generate a summary report.

As examples, two(2) snippets of the summary report in this work for the Teardrop and Butterfly curves are provided in Listing [3.5] and Listing [3.6], respectively.

The snippet only shows Report No. 09 specifically on Algorithm Execution Statistics. There are eight(8) prior reports in the full report summary list for each execution run.

Listing 3.5: Snippet of Teardrop Algorithm Summary Output

FC20-L0.18-A28-TEARDROP (REPORT NO. 09) ALGORITHM EXECUTION STATISTICS		
TEARDROP TOTAL-INTERPOLATED-POINTS	7.599000000000E+03	
TEARDROP SUM-CHORD-ERROR-(mm)	7.140807162860E-03	
TEARDROP SUM-CHORD-ERROR/TOT-INTERPOL-PNTS	9.398272128007E-07	
TEARDROP SUM-ARC-LENGTH-(mm)	1.018418663504E+02	
TEARDROP SUM-CHORD-LENGTH-(mm)	1.018418655699E+02	
TEARDROP DIFF-ARC-LENGTH-CHORD-LENGTH-(mm)	7.805327442156E-07	
TEARDROP PCNT-DIFF-ARC-CHORD-LENGTH	7.664163788301E-07	
TEARDROP SUM-CHORD-ERROR/SUM-CHORD-LENGTH	7.011661778683E-05	
TEARDROP SUM-ARC-THETA-(rad)	4.712268805770E+00	
TEARDROP SUM-ARC-AREA-(mm ²)	6.182290957317E-05	
TEARDROP SUM-ARC-AREA/SUM-CHORD-LENGTH	7.011661778683E-05	
TEARDROP AVG-CHORD-ERROR-(mm)	9.398272128007E-07	
TEARDROP AVG-ARC-LENGTH-(mm)	1.340377288107E-02	
TEARDROP AVG-CHORD-LENGTH-(mm)	1.340377277834E-02	
TEARDROP AVG-ARC-THETA-(rad)	6.201985793327E-04	
TEARDROP AVG-ARC-AREA-(mm ²)	8.136734610840E-09	
2023-10-05 10:43:45	Program run duration	19.907344569 seconds.

Listing 3.6: Snippet of Butterfly Algorithm Summary Output

FC30-L0.18-A28-BUTTERFLY (REPORT NO. 09) ALGORITHM EXECUTION STATISTICS		
BUTTERFLY TOTAL-INTERPOLATED-POINTS	1.234300000000E+04	
BUTTERFLY SUM-CHORD-ERROR-(mm)	4.846582536157E-03	
BUTTERFLY SUM-CHORD-ERROR/TOT-INTERPOL-PNTS	3.926902071104E-07	
BUTTERFLY SUM-ARC-LENGTH-(mm)	3.560730284349E+02	
BUTTERFLY SUM-CHORD-LENGTH-(mm)	3.560727930088E+02	
BUTTERFLY DIFF-ARC-LENGTH-CHORD-LENGTH-(mm)	2.354260789161E-04	
BUTTERFLY PCNT-DIFF-ARC-CHORD-LENGTH	6.611735799000E-05	
BUTTERFLY SUM-CHORD-ERROR/SUM-CHORD-LENGTH	1.361121273884E-05	
BUTTERFLY SUM-ARC-THETA-(rad)	2.211618559661E+01	
BUTTERFLY SUM-ARC-AREA-(mm ²)	1.298932073590E-03	
BUTTERFLY SUM-ARC-AREA/SUM-CHORD-LENGTH	1.361121273884E-05	
BUTTERFLY AVG-CHORD-ERROR-(mm)	3.926902071104E-07	
BUTTERFLY AVG-ARC-LENGTH-(mm)	2.885051275603E-02	
BUTTERFLY AVG-CHORD-LENGTH-(mm)	2.885049368083E-02	
BUTTERFLY AVG-ARC-THETA-(rad)	1.791945032946E-03	
BUTTERFLY AVG-ARC-AREA-(mm ²)	1.052448609294E-07	
2023-10-05 15:57:25	Program run duration	8.667691811 seconds.

3.27.2 Algorithm Functional Organization

The functional organization of software codes for the realtime interpolation algorithm in this work is shown in Figure [3.35].

The main algorithm in *src/algo* comprises a series of seven(7) calculation modules, with the last module being the writing of RS272/NGC G-Code. The name of each code file is self-explanatory.

The *src/common* and *src/cpp – codes* folders are for local and imported code utilities commonly shared among the modules. The Integrated Development Environment (IDE) used in this work is capable of compiling C, C++ and Ada source codes combined into a single binary executable.

The *src/curves* folder is where the ten(10) parametric curves in this work are located. The *src/files* folder is the directory where all source codes to generate data and reports are located. This includes code to generate the G-Code file.

The *src/parallel_pci* and *src/parallel_usb* are the directories containing interface codes for the hardware parallel port on the computer. The former hardware is for PCI Parallel Card interface, while the later hardware is for USB-to-Parallel device interface.

The *src/pthread* folder is for multi-threading codes (C/C++ and Ada) while the *src/realtime* folder is for Ada and C/C++ codes that implement realtime libraries. Note that Ada has its own built-in *Ada.Realtime* library.

Multi-threading in Ada is called *Tasking*. Multi-threading for the algorithm in this project was tested successfully, however the overall execution is too slow and so was abandoned. It was found that the creation and destruction of new threads for runtime parameter at every u-loop is time-consuming. With that, the serial and sequential mode of processing was adopted for the algorithm.

The *src/reports* folder is the storage location for all finished data and report files. The names of the files are meaningful like:

data_calc_frate_limit.txt,
data_calc_intgrl_error.txt,
data_calc_arclength_chordlength.txt,
data_calc_tang_accn.txt,
data_calc_time_lookahead.txt,
data_calc_u_next.txt,

data_calc_raw_curve.txt,

and many more. Depending on the total number of interpolated points for the run, each file size typically ranges between one to 10 megabytes.

The *obj* folder is the location of intermediate files during compilation, binding and linking. It is a system controlled folder and contains binary and semi-binary files. A snippet of the process of compilation, binding and linking is shown in Listing [3.7].

Finally, the *exec* folder is where the single binary executable for the algorithm in this work is located. This binary executable is less than 100 kilobytes, and is considered very small.

Figure 3.35: Algorithm-Functional-Components



,

Listing 3.7: Snippet of Algorithm Compilation, Binding and Linking

```
gprbuild -d -P/home/wruslan/WRY-UMP-Thesis/
/Teadrop-FC20-Lamda018-Algo28-Codes/c_multithreading.gpr -s -k

Compile
[C]          main_c_multithreading.c
[C]          c_position.c
[C]          c_velocity.c
[C]          c_accelern.c
[C]          calc03_feedrate_limit.c
[C]          calc00_parametric_curve.c
[C]          calc06_decide_next_frate.c
[C]          calc07_iterate_u_next.c
[C]          write_ngc_code.c
[C]          calc05_action_next_frate.c
[C]          calc01_lookahead_length.c
[C]          calc02_tang_accn_limit.c
[C]          calc04_integration_error.c
[C]          preempt_rt.c
[C]          main_usb-to-parallel-port.c
[C]          usb_parallel_port.c
[C]          pci_parallel_port.c
[C]          c_report_01.c
[C]          test_threads_01.c
[C]          c_min_max_int_dbl_in_array.c
[C]          c_dtstamp.c
[C]          c_random_int_dbl.c
[C]          c_parallel_port.c
Link
[archive]    libc_multithreading.a
[index]      libc_multithreading.a
[link]       main_c_multithreading.c
[2023-10-10 11:06:29] process terminated successfully, time: 02.36s
```

3.27.3 Algorithm and runtime parameters

The configuration of parameters in the realtime interpolation algorithm in this work is categorized into 3 sections, namely, the machine limits, the software runtime values, and the user specified runtime parameters. It is provided in Table [3.22] below.

Table 3.22: Algorithm and runtime parameters

	MACHINE LIMITS	VALUE	UNIT	DESCRIPTION
1	x_Vel_max	30.0	mm/s	X-axis maximum velocity
2	y_Vel_max	30.0	mm/s	Y-axis maximum velocity
3	x_Acc_max	30.0	mm/s ²	X-axis maximum acceleration
4	y_Acc_max	30.0	mm/s ²	y-axis maximum acceleration
5	Jerk_max	200.0	mm/s ²	Maximum allowable jerk
RUNTIME VALUES				
1	u_end_rise	0.05	nil	Rising S-curve range (0.0 .. 0.05)
2	rshape1	5.00	nil	Sigmoid curve rise shaping factor 1
3	rshape2	8.00	nil	Sigmoid curve rise shaping factor 2
4	u_start_fall	0.95	nil	Falling S-curve range (0.95 .. 1.00)
5	fshape1	5.00	nil	Sigmoid curve fall shaping factor 1)
6	fshape2	8.00	nil	Sigmoid curve fall shaping factor 2)
7	T_interpol	0.001	s	Interpolation time (period) 1 millisecond per step.
8	Error_tol	1E-6	mm	Chord-error tolerance for maximum allowable chord-error.
9	ngc_scale	1.0	nil	RS274/NGC G-code scaling factor
10	ngc_feedrate_min	3.0	mm/s	Minimum NGC feedrate setting for startup and shutdown of CNC machine.
11	PI constant used	PI	nil	PI = 3.14159_26535_89793_238 precision at 18 decimal digits (from internet)
RUN SPECIFIED				
1	Curve selection	Example AstEpi	nil	Curve selected for algorithm to execute. Ten(10) different choices for this work.
2	Feedrate Command FC	20.0	mm/s	Maximum feedrate set that should not be exceeded. For example: 10, 20, 30.
3	Lamda safety factor	0.18	nil	Acceleration limit safety factor. A number between (0.0 .. 1.0)

Note that the Feedrate Command (FC) is the value of the running feedrate that the user wants if it is following a straight line without any constraints. This is the user preferred feedrate.

The Feedrate Command is one of the four (4) components that determine the feedrate limit. The feedrate limit is dynamic, changes as parameter u changes, and is the minimum value among the four(4) components. The Feedrate Command is a user selected system constant. In the algorithm, the running feedrate is "forced and adjusted" to be as close to the feedrate limit throughout the parameter range.

3.28 Working environment setup

The four(4) computing machines in this work comprise HP-Laptop-01, HP-Laptop-02, HP-Desktop-01 and HP-Desktop-02. The two laptops are identical in hardware specifications (HP brand, 8-CPU cores, 16 GB Memory, 1.5 TB SSD storage and so on). Similarly, the two desktops are also identical in hardware specifications (HP brand, 8-CPU cores, 16 GB Memory, 1.5 TB SSD storage and so on). The complete hardware specifications for the computing machines in this work are provided in Table [3.23].

All four(4) computing machines are configured in a 3-way multiboot operating system. The user can select any one of the following operating systems: Linux Debian 10 Buster, Linux Ubuntu 20.04 LTS, or Microsoft Windows 10 Professional. The details are provided in Table [3.24].

All four(4) computing machines are installed with the same application software. This means the software installed are clones of each other. The details are provided in Table [3.25] for the programming languages, Table [3.26] for the reporting software, and Table [3.27] for the specialized software applications.

In the table for specialized software applications, the LinuxCNC-Axis software is the primary application to drive the CNC machine for all the G-codes generated in this work. The Panaterm application is used to communicate and monitor the status of the proprietary CNC-Servo drives responsible for control of the CNC machine. The oscilloscope software, available for both Linux and Windows versions, are used to simultaneously trace 8-channels of digital or analog electrical signals. The CuteCom application is an RS232 serial communication software used to drive and monitor serial signals in this work, including snooping, that is, a third party listening to RS232 serial communication between two serial devices. The DAQNavi Control Application is for Advantech products like PCIE-1884 DAQ card, and for user interface development using Qt6 C/C++ codes.

3.29 System Hardware Environment

Table 3.23: System Hardware Specifications

No	Category	Specification Description
1	HP-Laptop-01	Model Hewlett-Packard, HP EliteBook 8570w
2	HP-Laptop-02	Intel(R) Core(TM) i7-3630QM CPU @ 2.40GHz CPU Two Quad(4) cores, 64 bits, with 8 threads DDR3 16 GB System board memory, 1600 MHz clock System Caches: L1 32 KB, L2 256 KB, L3 6 MB SDDs: 1TB and 512GB Kingston Solid State Drives Display NVIDIA Corporation GK107GLM GPU NVIDIA Corporation [Quadro K1000M] Parallel parport0 PC-style 0x378 (0x778), irq 5
3	HP-Desktop-01	Model Hewlett Packard EliteDesk 800 G1 TWR
4	HP-Desktop-02	Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz CPU Two Quad(4) cores, 64 bits, with 8 threads DDR3 18 GB System board memory, 1600 MHz clock System Caches: L1 256 KB, L2 1 MB, L3 8 MB SDDs: 1TB and 512GB Kingston Solid State Drives Display NVIDIA Corporation GK208 64bits 33Mhz GPU NVIDIA Corporation [GeForce GT 710B] PCI parport0: PC-style 0xd100, irq 16 [PCSPP,TRISTATE] Advantech PCIE-1884 Signal processing card 32 bits

3.30 Software Environment

3.30.1 Operating System

Table 3.24: Operating Systems

No	Category	Specification Description
1	Ubuntu 20.04.6 LTS	HP-Desktop-01 and 02, HP-Laptop-01 and 02 Codename: focal, LSB version core-11.1 Kernel: linux-5.15.0-83-lowlatency SMP PREEMPT (2023) x86_64 Preemptive Realtime OS, Symmetric Multi-Processor
2	Debian GNU/Linux 10	HP-Desktop-01 and 02, HP-Laptop-01 and 02 Codename: Buster Kernel: 4.19.0-25-rt-amd64 x86_64 GNU/Linux SMP PREEMPT RT Debian 4.19.289-2 (20230808) Preemptive Realtime OS, Symmetric Multi-Processor
3	MS Windows 10 Pro	HP-Desktop-01 and 02, HP-Laptop-01 and 02

3.30.2 Programming Software Languages

Table 3.25: Programming Software Languages

No	Category	Specification Description
1	C/C++, Ada	GNAT Studio Community 2021 (20210423) GNAT 9.4.0 targeting x86_64-linux-gnu SPARK Community 2021 (20210519)
2	GUI for C/C++	Qt Creator 10.0.1 x86-64 Based on Qt 6.4.3 (GCC 10.3.1 20210422) Built on May 04, 2023
3	Python	Python 2.7.18 (Jul 1 2022), [GCC 9.4.0] on linux2 Python 3.8.10 (May 26 2023), [GCC 9.4.0] on linux3
4	Julia/Atom	Julia Version 1.9.0 (2023-05-07)/Atom 1.58.0 x64

3.30.3 Reporting Software Applications

Table 3.26: Reporting Software Applications

No	Category	Specification Description
1	TeXstudio	TeXstudio 3.0.1 (git n/a) Using Qt Version 5.12.8, compiled with Qt 5.12.8 R
2	LibreOffice Suite	Version: 6.4.7.2, Build ID: 1:6.4.7-0ubuntu0.20.04.8 CPU threads: 8; OS: Linux 5.15; UI VCL: gtk3;
3	Master PDF Editor	Version 5, Build 5.7.60, 64 bit, (2021) GCC: 5.3.1, GLIBC: 2.17, Qt: 5.9.5, SANE: 1.0.25
4	Dia diagram editor	Dia Version 0.97 + git (2011)
5	Gnuplot	Gnuplot Version 5.2 patchlevel 8 (2019-12-01)

3.30.4 Specialized Software Applications

Table 3.27: Specialized Software Applications

No	Category	Specification Description
1	CNC Control App	LinuxCNC/Axis ver. 2.8.0 (2016) on Linux Debian 10 (Buster)
2	CNC Monitoring App	Panaterm version 4.5 (Panasonic) on Microsoft Windows 10 Pro
3	Oscilloscope App	Bitscope Digital Storage Oscilloscope (DSO) Linux - Version 2.8 Intel x86-64-bit Windows - Version 2.8 Intel x86-64-bit
4	Serial Communication RS232 Snooper circuit	CuteCom graphical serial terminal (RS232) Linux GUI Version 0.30.3 (2015) Windows GUI Version 0.30.3 (2015)
5	DAQNavi Control App	Advantech PCIE-1884 Data Acquisition Card Linux GUI Version 4.0.8.0 64-bit Windows GUI Version 4.0.8.0 64-bit

3.31 CNC System Setup

The image of the CNC machine used in this work is shown in Figure [3.36]. The next Figure [3.37] shows the three(3) Panasonic AC-Servo-Drives for the x, y, and z axes motions of the CNC machine. For this machine, the z-axis is used to move the pen tool up and down.

The next image shown in landscape mode in Figure [3.38] illustrates the operations of the LinuxCNC-Axis software in driving the CNC machine to draw the diagram on its screen. The diagram to be drawn is represented by a G-code, a snippet of the code is provided in Listing [3.8].

The image of the CNC system environment is shown in Figure [3.39] and Figure [??]. *Note: The images will be changed later.*

The overview of the CNC system environment is shown in Figure [3.40]. The multiplexed monitoring of the CNC Panaterm controller is shown in Figure [3.41]. The wiring diagram connections for parallel port, CNC servos and PCIE-1884-card is shown in Figure [3.42]. The snooper circuit for a third party computer to listen to RS232 serial communications between two devices is provided in Figure [3.43].

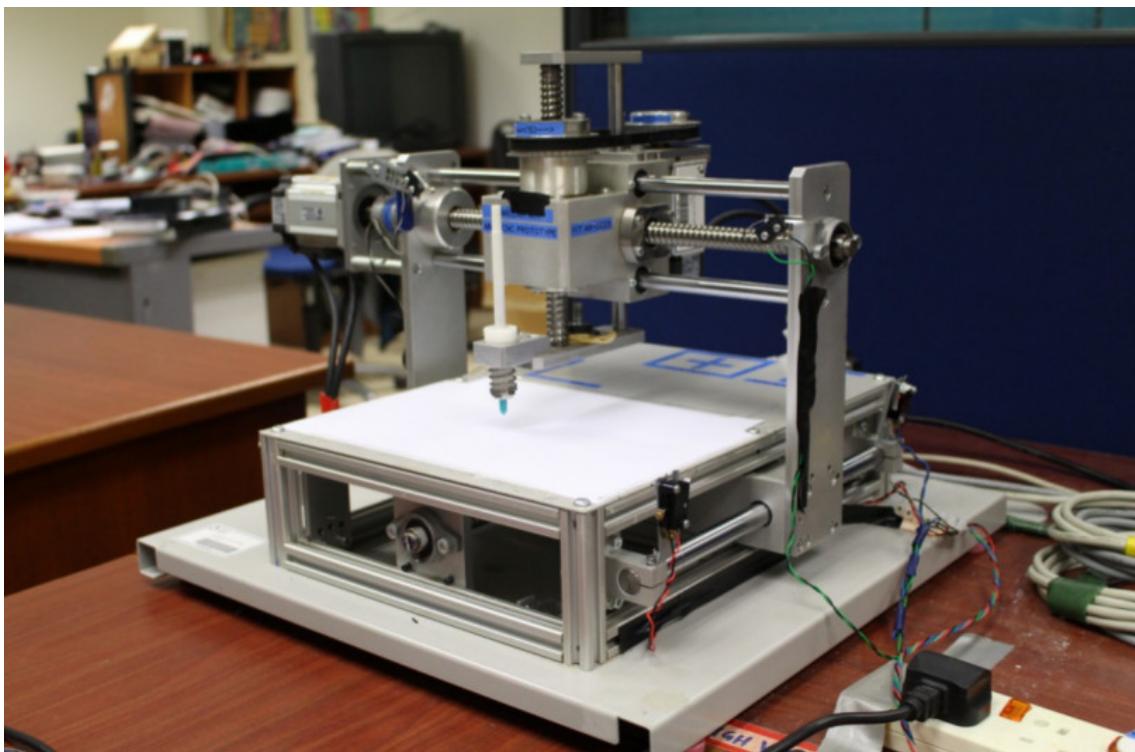


Figure 3.36: The prototype 3-axis CNC research machine

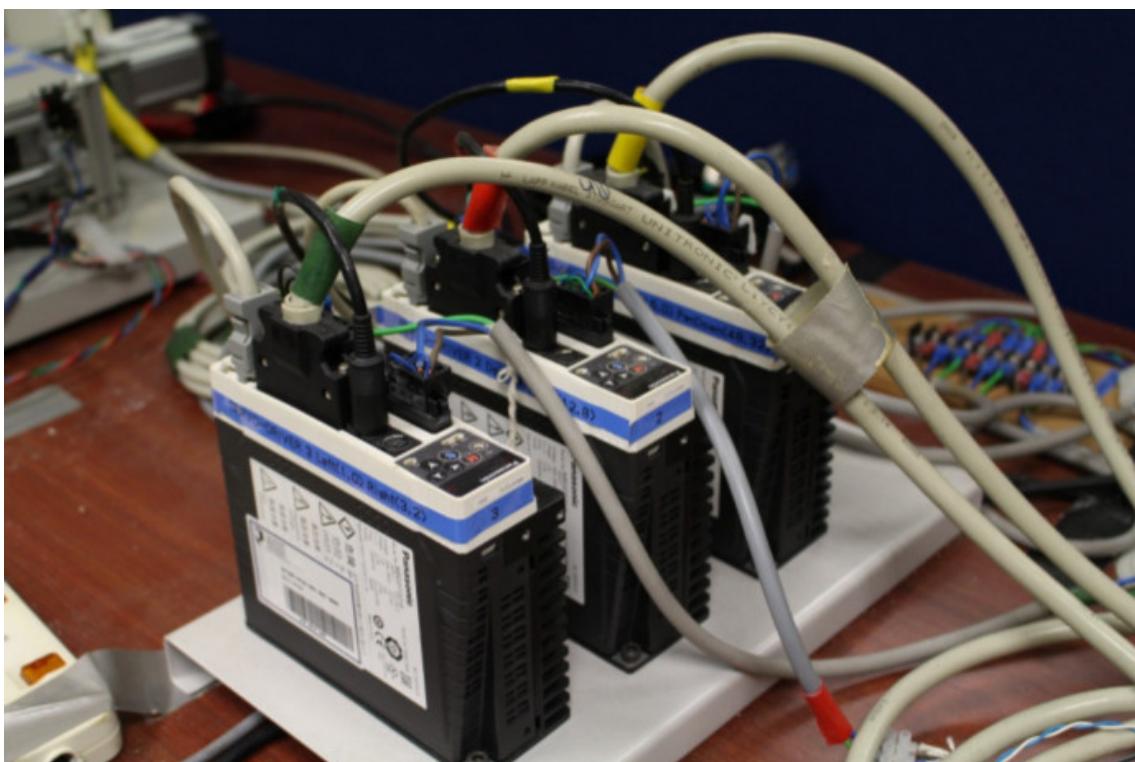
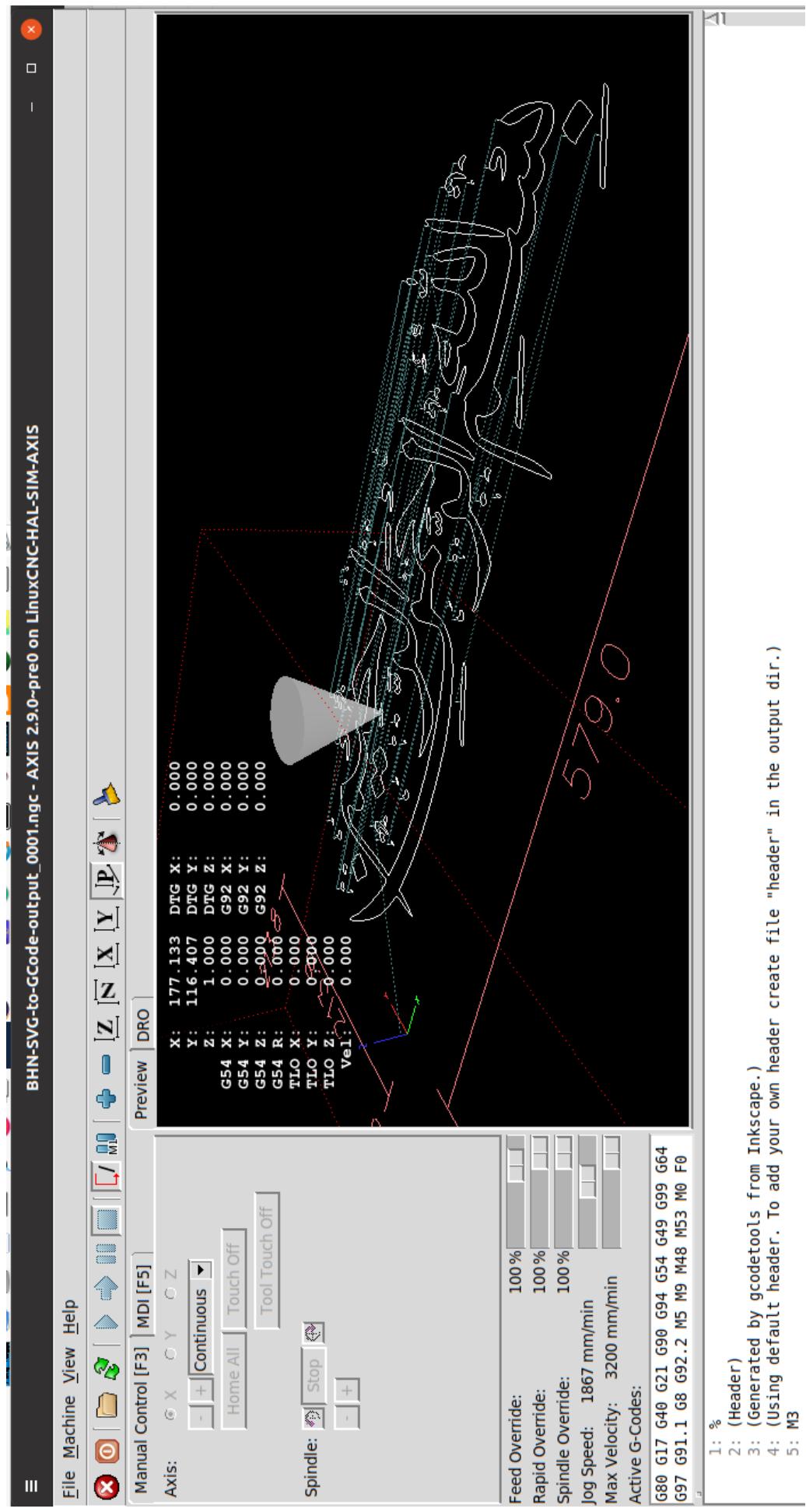


Figure 3.37: The 3-sets AC servo drivers for the CNC research machine

Figure 3.38: Validation of G-code using LinuxCNC Axis software



Listing 3.8: G-code snippet for Arabic calligraphy

```
(Header)
(Generated by gcodetools from Inkscape.)
(Using default header. To add your own header create file "header" in the output dir.)
M3

(Header end.)
G21 (All units in mm)
(Start cutting path id: path11762)
(Change tool to Default tool)
G00 Z5.000000
G00 X235.600000 Y35.100000
G01 Z1.000000 F100.0(Penetrat)
G03 X213.627800 Y25.129879 Z1.000000 160.683117 J-162.930013 F400.000000
G03 X212.000000 Y22.400000 Z1.000000 11.475151 J-2.729879
G03 X212.577899 Y21.936846 Z1.000000 10.474545 J-0.000000
G03 X218.800000 Y23.900000 Z1.000000 1-8.264838 J37.036796
G02 X225.555826 Y26.585083 Z1.000000 1199.253827 J-491.492772
G02 X241.000000 Y32.500000 Z1.000000 1527.010173 J-1352.932266
G03 X257.708311 Y40.036826 Z1.000000 1-45.059557 J122.180722
G03 X259.000000 Y42.200000 Z1.000000 1-1.165476 J2.163174
G03 X258.293460 Y42.724778 Z1.000000 1-0.548158 J0.000000
G03 X235.600000 Y35.100000 Z1.000000 1111.496791 J-369.428872
G01 X235.600000 Y35.100000 Z1.000000
G00 Z5.000000
...

```

Figure 3.39: The front end environment of the system



Figure 3.40: Overview CNC system environment

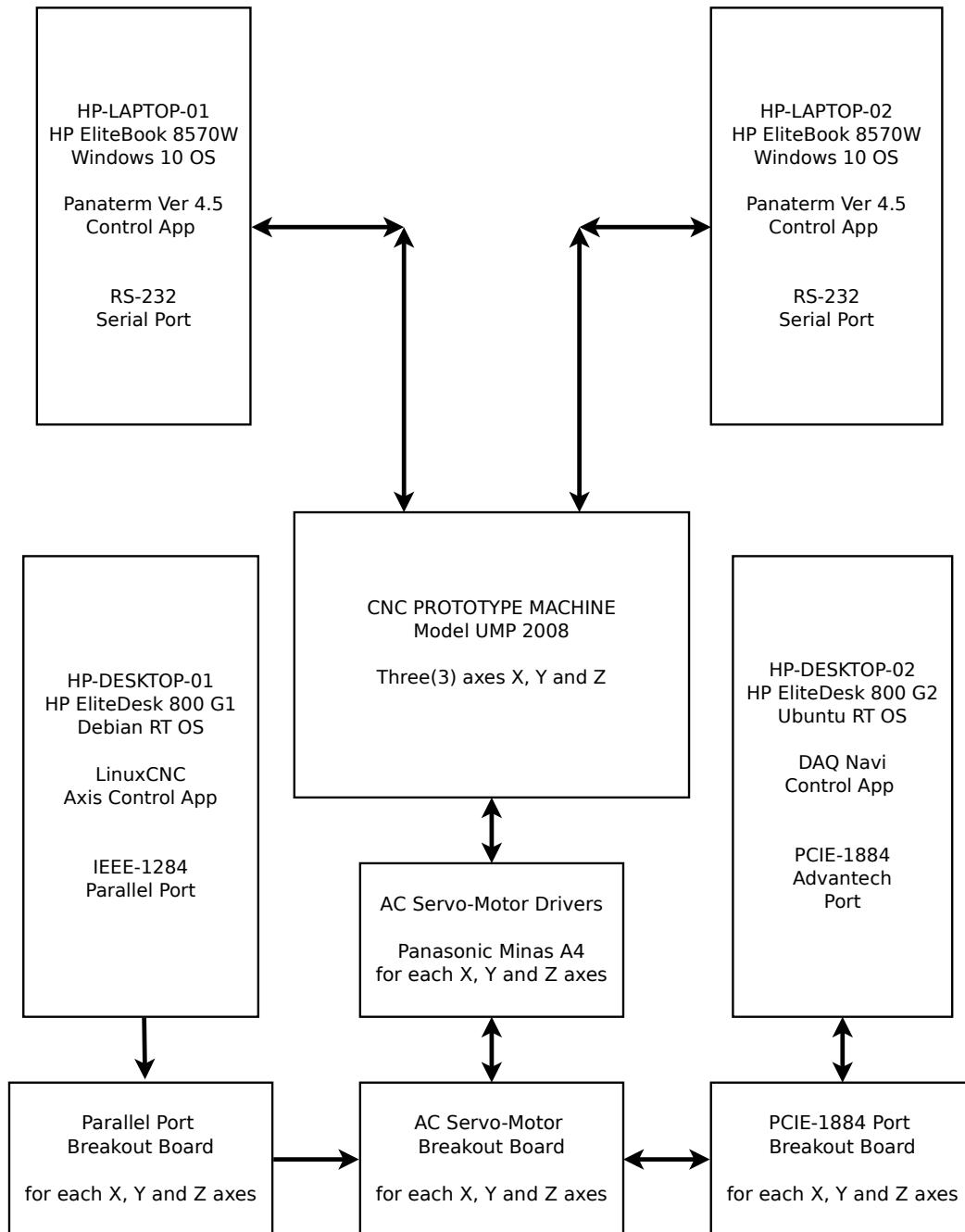


Figure 3.41: CNC-system-Panaterm-controller

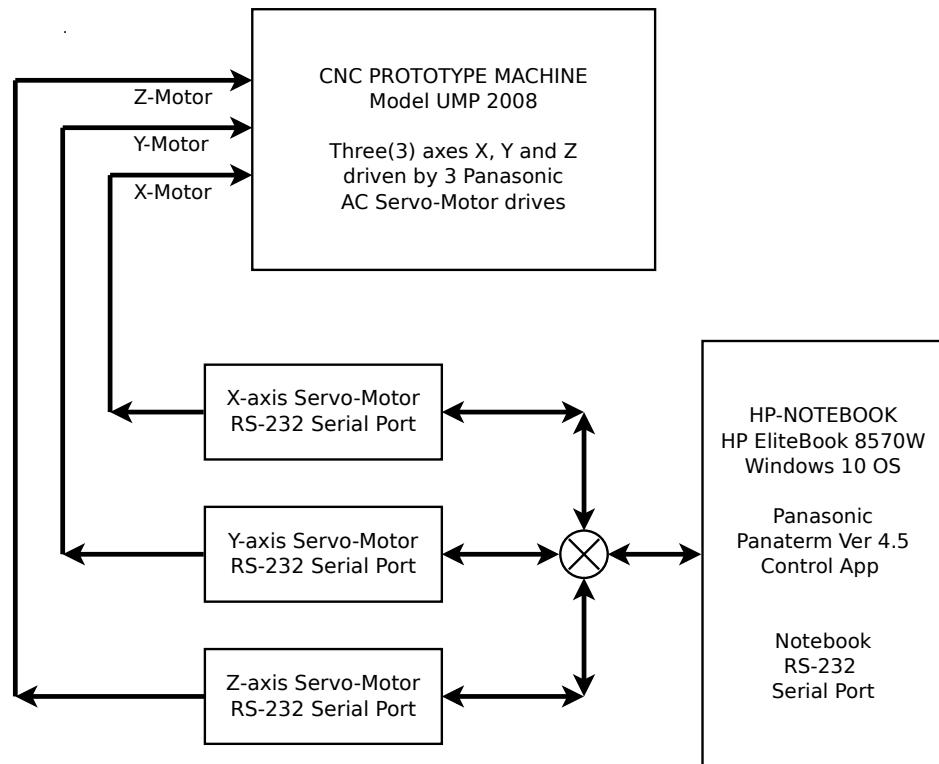


Figure 3.42: Parport-CNC-servo-PCIE-1884-wiring-diagram

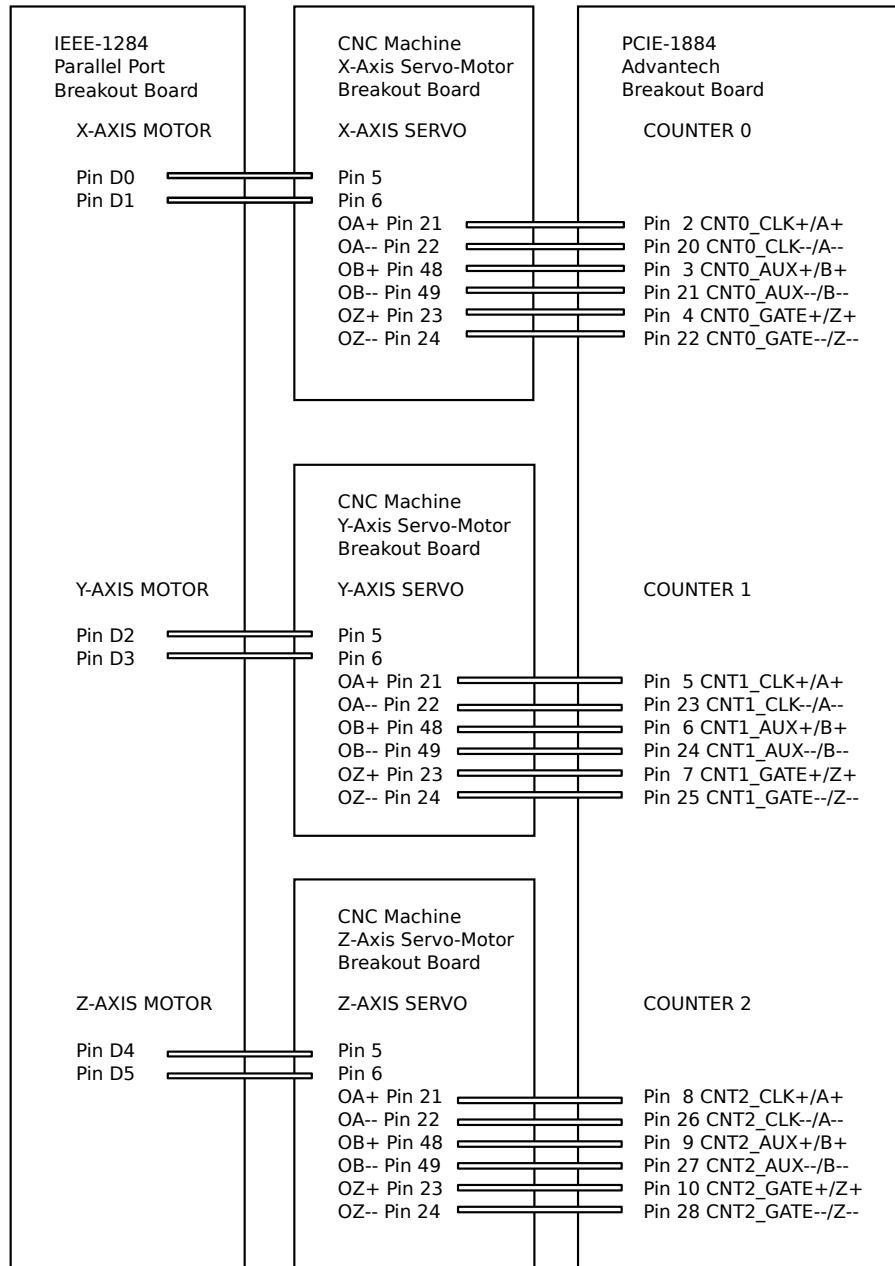
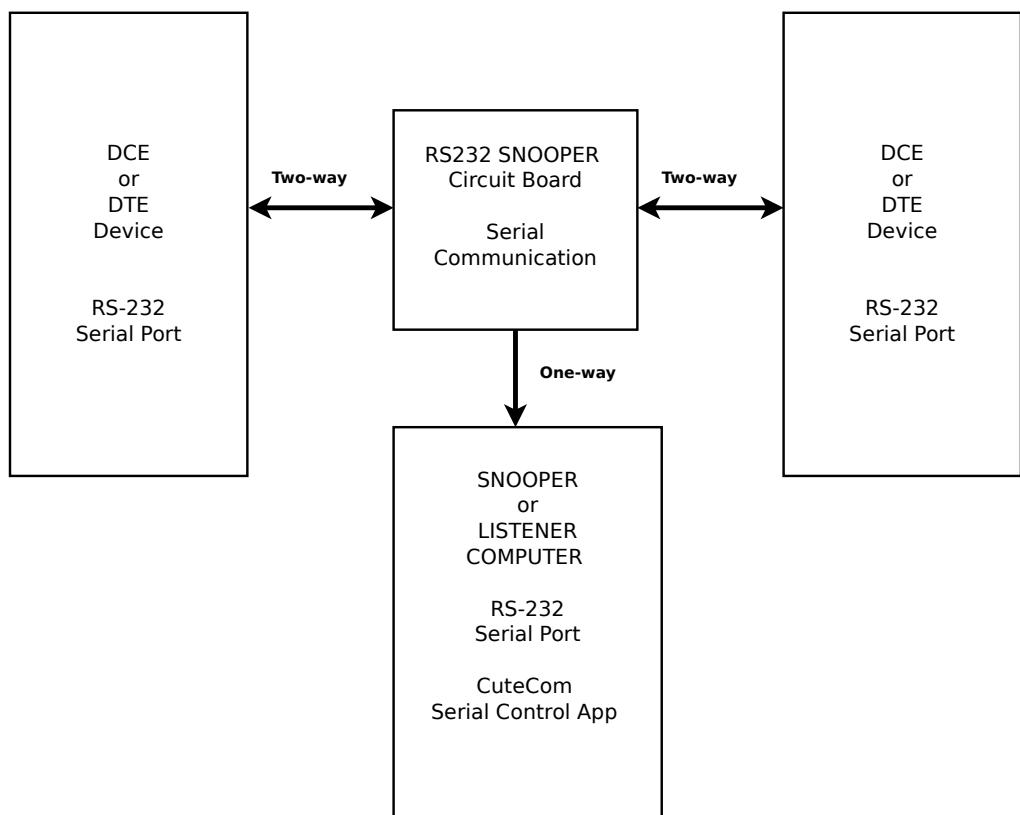


Figure 3.43: RS232 Snooper Circuit



DCE = Data Communication Equipment

DTE = Data Terminal Equipment

Chapter 4

RESULTS AND DISCUSSIONS

4.1 CHAPTER ORGANIZATION

4.1.1 Result comparisons with Published Reference Paper

This section discusses the results of this work in comparison to the published reference paper by Zhong et al. (2018), titled "A realtime interpolator for parametric curves". The results in this work showed similar patterns and comparable values to the published referenced results. The comparisons cover the feedrate variations along the curve, the velocity profiles of the x and y axes motions, the current running feedrate profile against the feedrate-limit, and the acceleration profile. These results spanned the entire traversal along the parametric curve.

4.1.2 Algorithm Executions

This section discusses the total number of algorithm executions, history of algorithm revisions, selection of an illustrative curve among the ten(10) parametric curves, acceleration jitters, and determination of the acceptable or nominal value of the acceleration safety factor lamda to avoid acceleration jitters.

4.1.3 Teardrop Curve for Illustration

This section describes the characteristics of the Teardrop curve, and the basic reasons for making the Teardrop curve as an illustration curve. The rest of the nine(9) curves are described in comparison to the Teardrop curve.

4.1.4 Results of Teardrop curve

This section discusses the full outputs of the Teardrop curve resulting from the algorithm execution. Similar outputs for the rest of the nine(9) curves are provided in their respective appendices.

These illustrative outputs cover, for example, the direction of travel for the curve, the radius of curvature $\rho(u)$, the first and second-order terms in Taylor's expansion, the chord-error absolute constraint, the running feedrate absolute constraint, the four(4) components of the feedrate limit, histogram distribution of interpolated points, starting and ending feedrate profiles, color coded running feedrates and many more.

4.1.5 Notable Results for Rest of Curves

This section discusses specific aspects of the execution results worthy of further clarification. Four(4) special cases were considered. Results of the Circle and Ellipse curves that can be used as a means of validation and verification of algorithm performance were highlighted. Results that look abnormal but are correct need to be explained. This cover negative results for the difference (SAL-SCL) in the Teardrop curve, and machine epsilon problems for the SnaHyp curve.

4.1.6 Overall Execution Results

This section presents the algorithm execution summary results for all of the ten(10) parametric curves studied in this work. The discussions include the Feedrate Command (FC) and the Lamda safety factor (LSF), algorithm validation and verification (V & V),

LinuxCNC-Axis simulation run validation (SRV), real run validation on the CNC machine (RRV), and algorithm performance measures.

The performance measures cover algorithm generated variables like total interpolated points (TIP), total sum-arc-length (SAL), total sum-chord-length (SCL), total sum-chord-error (SCE), total sum-arc-theta (SAT), and total sum-arc-area (SAA).

The four(4) assessment metrics for algorithm performance are: (SCE/TIP), (SCE/SCL), (SAA/SCL) and $100*(SAL-SCL)/SAL$. The first three terms are ratios while last term is a percentage measure of the difference.

4.1.7 Summary of Results Chapter

This section summarizes the important findings in this chapter.

IMPORTANT NOTE: Some data tables and graphs are displayed in landscape mode, when it is not possible in portrait mode. The landscape mode facilitates presentation of multiple plots side by side, including and especially wide data tables. This mode helps comprehension, visualization and comparison of information.

4.2 COMPARISONS AGAINST PUBLISHED PAPER

4.2.1 Result comparisons with Published Reference Paper

It is important to note that the comparisons of results in this work can only be made against the Teardrop parametric curve in the referenced paper by Zhong et al. (2018), because this work uses the same parametric equation. And it must also be reminded that the single interpolation algorithm designed and developed in this work is to cater ten(10) different parametric curves of various features, shapes and dimensions, not just the one simple Teardrop curve.

The computational algorithms are also different between this work and the published paper. The overall algorithm in this work is considered simpler (not complicated branching flow), highly structured (for example, one-way-in and one-way-out for every computing unit), and therefore easy to understand and maintain. In addition, this algorithm strictly enforces complete and simultaneous non-violation of chord-error and feedrate-limit constraints. The results is absolute success when applied to all ten(10) different parametric curves.

For the above reasons, it cannot be said that the algorithm is an-apple-to-apple comparison with that of the published paper, even for the identical Teardrop curve. Obviously, the computation techniques are different. However, the common theme is that both are "interpolation algorithms", meaning given a current point on the parametric curve, find the next point for the move on the curve. These are the reasons it was stated that the results in this work showed similar patterns and comparable values. The comparisons are described in the next section.

4.2.2 Display comparison results

The four(4) comparisons for the Teardrop curve cover :

- (1) Feedrate variations along the curve shown in Fig[4.1].
- (2) Velocity profiles of the x and y axes motions shown in Fig[4.2].
- (3) Running feedrate profile against the feedrate-limit shown in Fig[4.3].
- (4) Acceleration profile containment shown in Fig[4.4].

4.2.3 Discussion on comparative results

It is important to note that This-Work (PhD thesis) provides results that are similar and comparable to the work produced in the Reference-Paper. This comparison is only applicable to the Teardrop because both adopted the same exact equation for the parametric curve. The four(4) figures for comparison in the next section are provided in landscape mode. The legend colors of the curves displayed in the figures were made identical for easy visual comparison.

In Fig[4.1], the feedrate variations in the interpolation are similar and comparable. The generally lower feedrate values in This-Work as the (x, y) position point moves along the Teardrop trajectory is due to both the simultaneous absolute constraints on chord-error (epsilon) and non-violation of the machine feedrate limit. In the Reference-Paper, there is a push-up of the running feedrate, which causes an increase in chord-length, thereby an increase in chord-error. In the Reference-Paper there is no specific mention of absolutely restricting the chord-error for each interpolation point in parameter (u).

Also in Fig[4.1], the different colors refer to the magnitudes of the feedrate (mm/s), which varies from point-to-point on the parametric curve. In this work, the colors transition smoothly following the color spectrum. There is no jumping of colors. The direction of travel of the (x, y) point is starting from the apex going counter-clockwise.

In Fig[4.2], the velocity profiles of the x and y axes motions are also similar and compa-

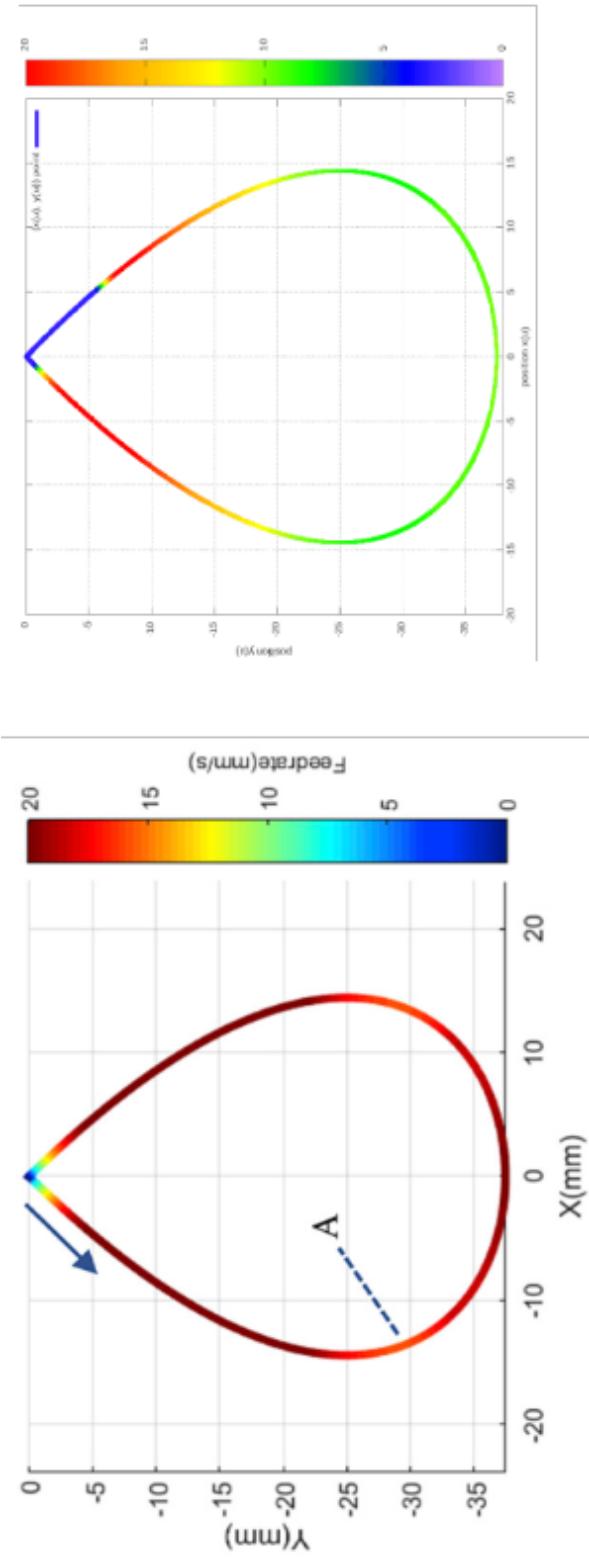
table. However, the number of interpolated points in This-Work (7599 points) is higher than the Reference-Paper. See the x-axis for the runtime in seconds. Each cycle of interpolated points is set the same for both works, that is, at 0.001 (s) or one millisecond. This makes the completion time for the interpolation algorithm in This-Work as 7.6 seconds, while that for the Reference-Paper as about 6.3 seconds.

In Fig[4.3], the actual running feedrate (velocity) almost overlaps with the machine feedrate limit calculated by the algorithm. The feedrate determination is one of the most important and difficult computation to handle. It is a combined effect of both dynamical and kinematical constraints which varies among different CNC machines.

Also in Fig[4.3], with the exception of the initial rise and final fall of the actual feedrates, the middle section shows that in both the Reference-Paper and This-Work, the actual running feedrate follows exactly the value of feedrate limit calculated by the algorithm. In this middle region, it is almost overlapping. The actual initial rise and final fall of the actual feedrates are user-selected S-curves. In This-Work, the initial S-curve-rise region is set at ($u = 0.00$ to 0.05) and the final S-curve-fall region is set at ($u = 0.95$ to 1.00). In the Reference-Paper, the characteristics of both S-curve-rise and S-curve-fall regions are not specifically mentioned.

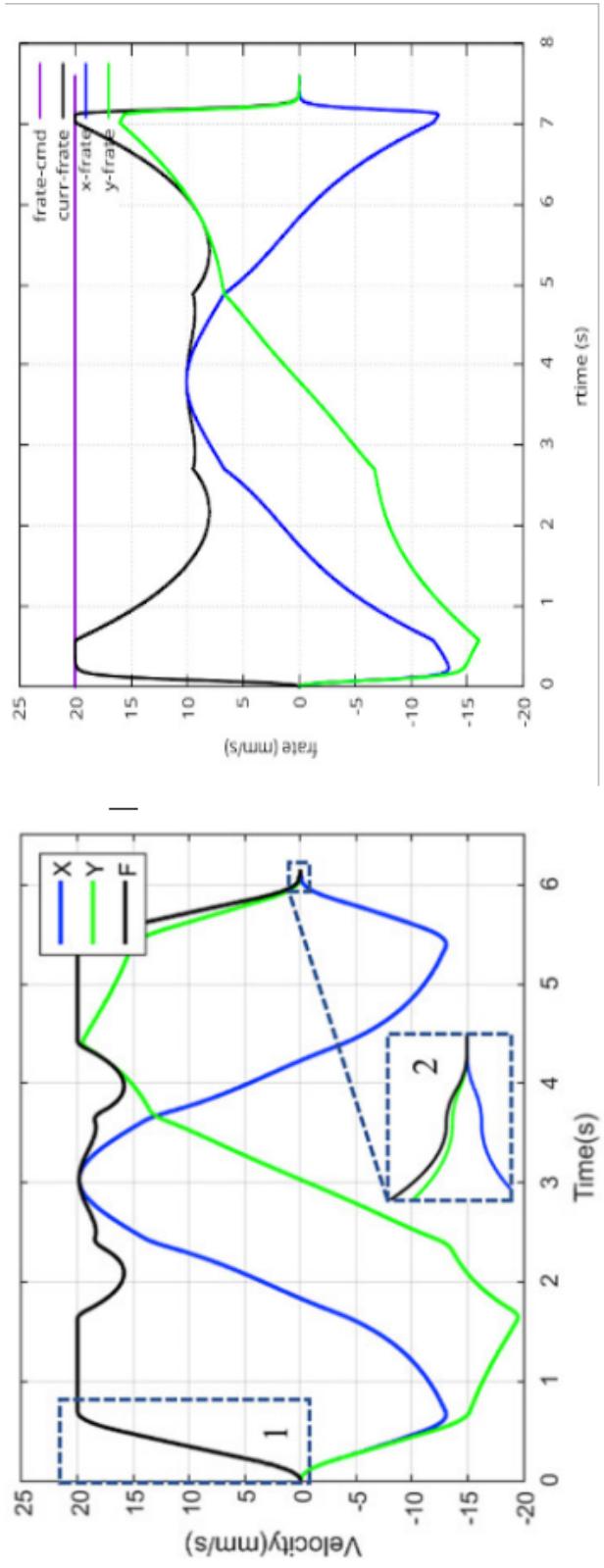
In Fig[4.4], the actual running tangential acceleration is constrained within the minimum and maximum accelerations in both the Reference-Paper and This-Work results. The acceleration values are not identical but comparable. The feature for acceleration constraints (min, max) is similar. In This-Work, the value of lambda (acceleration safety factor) was selected to be 0.18, chosen to eliminate acceleration jitters that causes machine velocity jerks. This number 0.18 is good when applied to all ten(10) different parametric curves covered in This-Work. In the Reference-Paper, it was only mentioned that lambda should be within (0.00 and 1.00). The determination of acceptable lambda (acceleration safety factor) is discussed in detail in Section [4.3.4] in this chapter.

Figure 4.1: Feedrate variations along the curve



Equivalent legend: Reference-Paper (left figure) versus This-Work (right figure)

Figure 4.2: Comparisons velocity profiles of the x and y axes motions



Equivalent legend: Reference-Paper (left figure) versus This-Work (right figure)

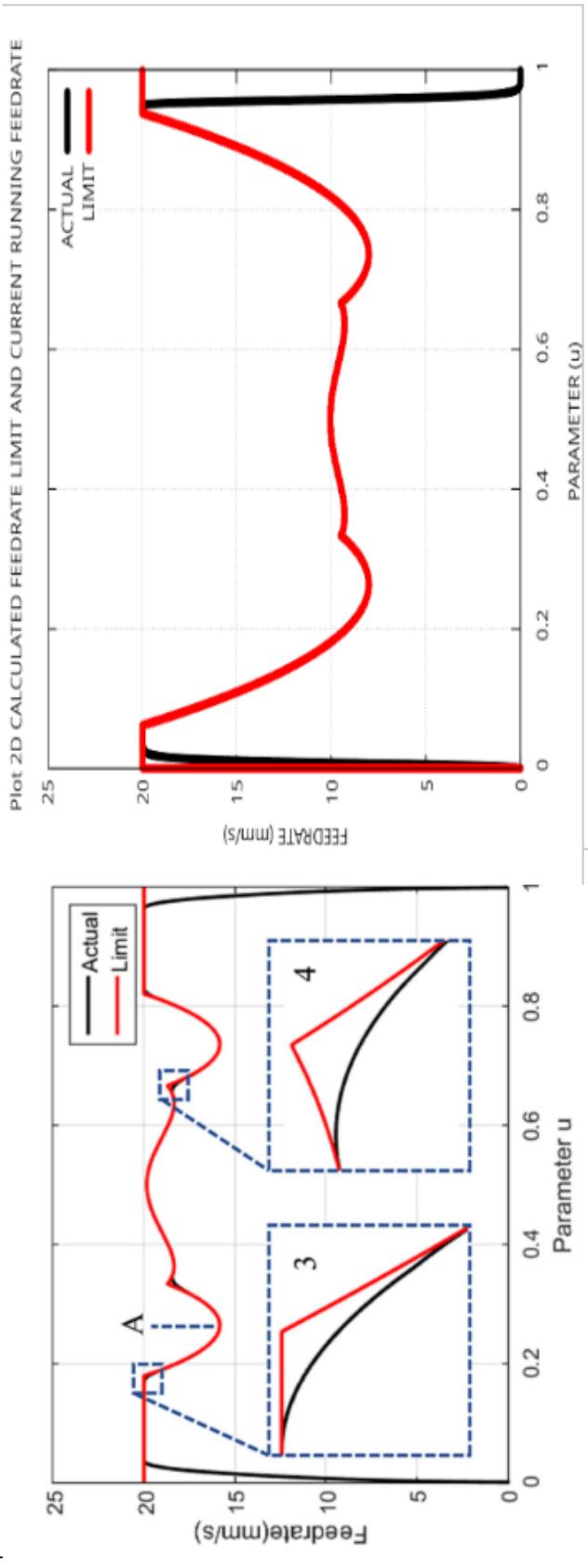
F = curr-rate (Current running feedrate)

X = x-rate (X-axis feedrate)

Y = y-rate (Y-axis feedrate)

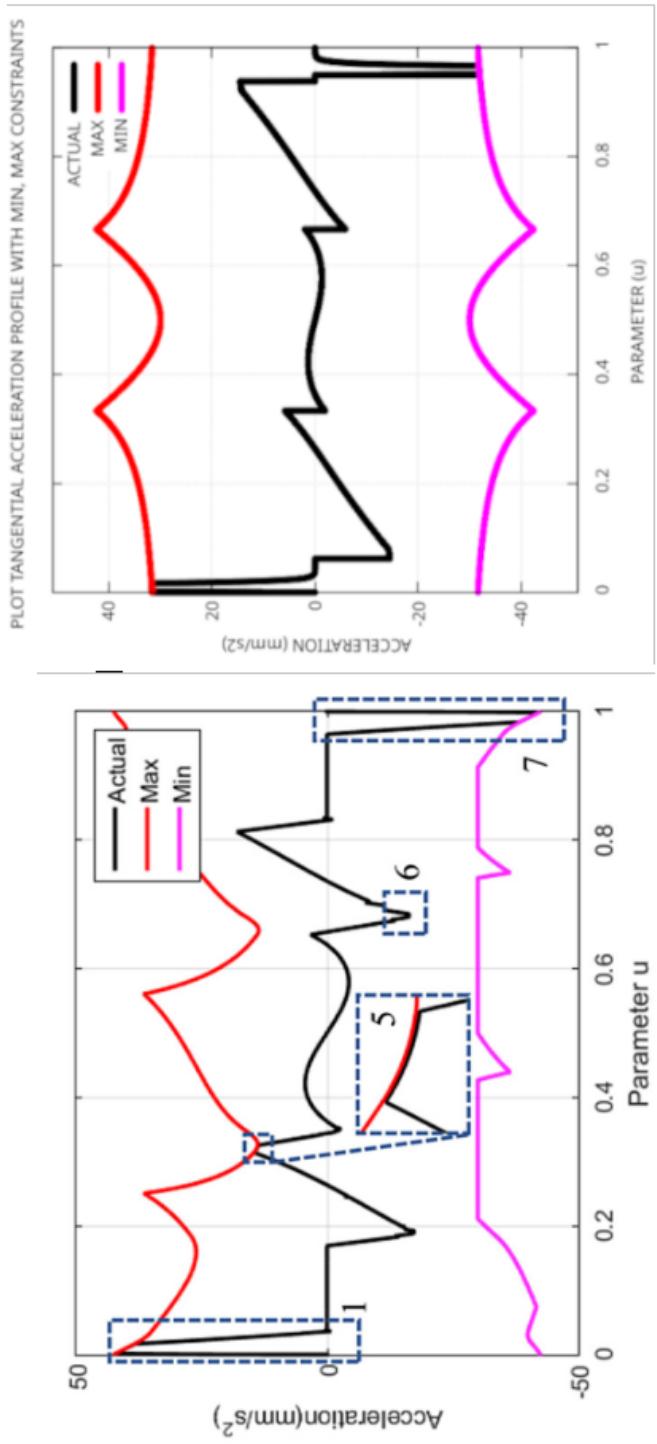
Time = rtime (Runtime)

Figure 4.3: Running feedrate profile against the feedrate-limit



Equivalent legend: Reference-Paper (left figure) versus This-Work (right figure)

Figure 4.4: Acceleration profile containment



Equivalent legend: Reference-Paper (left figure) versus This-Work (right figure)

4.2.4 Validation of Teardrop curve on CNC machine

The following are runtime characteristics of the Teardrop curve computed by the interpolation algorithm for FC20, the Feedrate Command and Lamda = 0.18, the acceleration safety factor.

1. Width (mm) = 28.8
2. Height (mm) = 37.5
3. Total interpolated points = 7599
4. Sum Total arc length (mm) = 101.8418663504
5. Sum Total chord-length (mm) = 101.8418655699
6. Sum Total chord-error (mm) = 0.007140807163
7. Average chord-length (mm) = 0.013403772778

Notice that the CNC machine moves about 7600 incremental steps to cover the distance of 102 mm (about 4 inches) of the total Teardrop curve. This step move is very small indeed, maintaining chord-error below 1 (nm) nanometer and speed (feedrate) below the feedrate limit throughout the entire curve. The average chord-length or linear move per step is 0.0134 (mm). It is quite remarkable that the simple CNC machine is capable of the very small increments.

A snapshot of the CNC machine in operation is shown in Fig[4.5] on the next page. A simple pen is used to trace the trajectory of the Teardrop curve. The results for the completed execution of the Teardrop curve is shown next in Fig[4.6]. The run was repeated twice for confirmation of correctness.

Figure 4.5: Execution of Teardrop Curve on CNC Machine

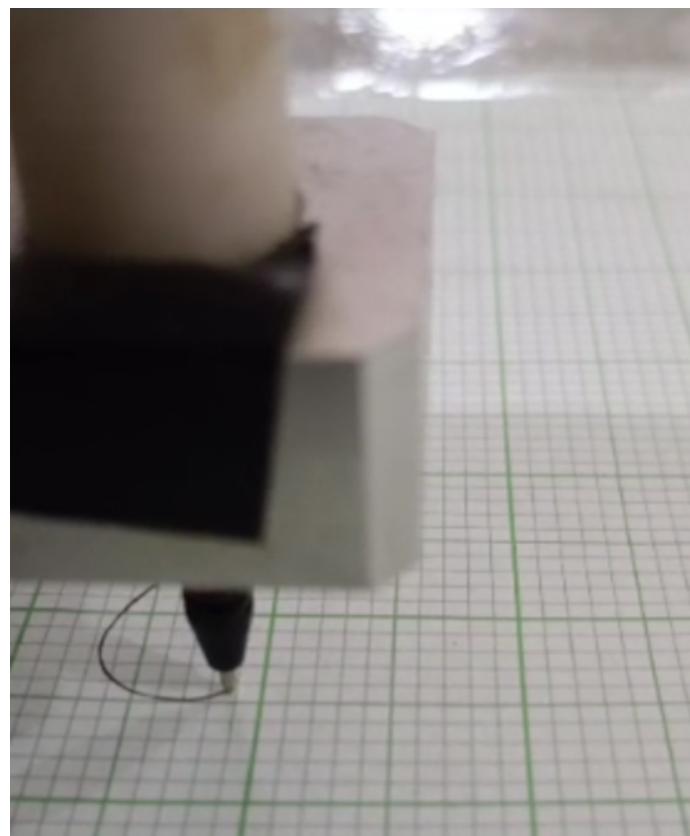
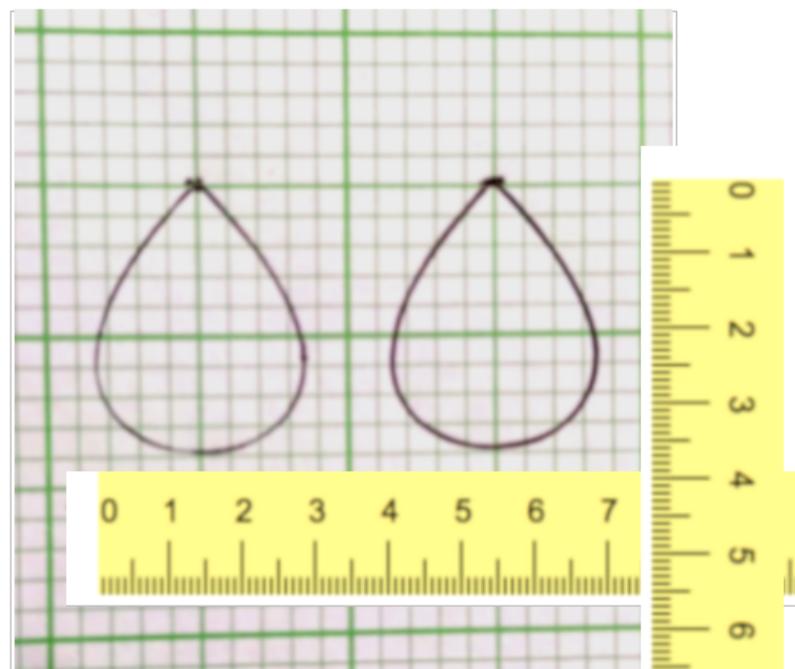


Figure 4.6: Results of Teardrop Curve CNC Machine execution - ruler (cm)



4.3 ALGORITHM EXECUTIONS

4.3.1 Total algorithm executions

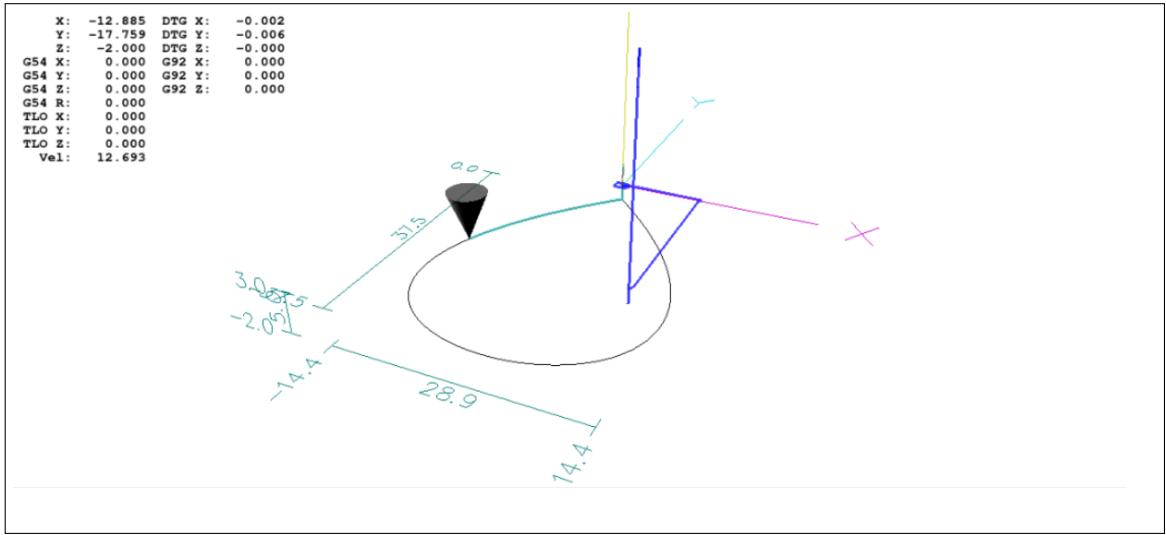
In this work we have executed the algorithm for ten(10) different curves, four(4) different feedrate commands (FC 10, 20, 30, 40), and four(4) different Lamda safety factors (Lamda 0.10, 0.18, 0.20, 0.50). This combination makes the total number of algorithm executions at 160. For each execution, 3 different input parameters are required, that is, curve type selected, feedrate command FC, and Lamda safety factor.

It is expected that a variety of issues arise in these executions due to the diverse characteristics and sizes of the ten(10) curves, the different running feedrates against the machine limits, and the most suitable value of the Lamda safety factor. These interesting issues will be discussed in the forthcoming sections.

4.3.2 Algorithm revisions

The current revision of the realtime interpolation algorithm is major version 28. The algorithm version 1 started in June, 2022. The software architecture was completed at version 10 by September, 2022 (3 months). Software coding implementation in C/C++ programming language was completed by version 15 in December, 2022 (3 months). Unit and system testing were finished in version 20 by March, 2023 (3 months). Writing major program reports, plotting, recording execution to files, took another 3 months until June, 2023, arriving at version 25. Finally, the full execution of 160 runs, with minor revisions along the way, ended up with the 28th version. This is the state of the software as of October, 2023.

Figure 4.7: Teardrop Perspective View 3D in LinuxCNC-Axis



4.3.3 Illustrative example Teardrop curve execution

The above figure shows a real live execution of the Teardrop parametric curve by the algorithm developed in this work. The job of the interpolation algorithm is to generate successive points along the curve trajectory so that the CNC cutting tool (laser cutter) illustrated by the cone in the figure follows the path accurately. The curve begins at parameter $u = 0.00$ (starting point), increasing in steps until $u = 1.00$ (ending point), as the entire Teardrop curve is being followed, that is, from start to finish. References: Zhong et al. (2018), and B Shengzhou and Jiang (2020)

The algorithm uses the second-order Taylor's approximation to calculate the steps ($u\text{-next}$) in parameter u , and at the same time constrains both the chord-error (deviation from the true curve path) to below a set error tolerance (1E-6 mm), and the running feedrate to be very close but below the feedrate limit throughout the full curve path.

The feedrate limit at every parameter u point is calculated by the algorithm based on geometrical, dynamical and kinematical constraints. The constraints comprise 4 different components: user set Feedrate Command FC, minimum and maximum CNC machine axial velocities, minimum and maximum CNC machine axial accelerations, and the geometric factors of the curve path like bends and sharp turns.

The main objective of the algorithm execution is to ensure that the resulting running feedrate (speed motion of the cutting tool) is smooth and continuous, and not exceeding the feedrate limit throughout the full curve path. Note that the feedrate limit varies with u , and thus, the running feedrate also varies, for example, when negotiating curves and sharp bends. The algorithm accomplishes the tool motion strictly without violating both the chord-error and running feedrate constraints.

Since the smoothness of running feedrate is critical to the success of the algorithm, any acceleration jitters (rapid acceleration fluctuations) will result in jerky machine feedrates. This situation is not acceptable. The next section discusses lamda, the acceleration safety factor, and how the algorithm handles this important subject.

4.3.4 Determination of acceptable lamda

In order to execute the interpolation algorithm three(3) user inputs are required, namely: curve type, feedrate command FC, and lamda safety acceleration factor.

The feedrate command FC, is a user specified value that must be within acceptable limits of the CNC machine. The Table [3.27.3] in Chapter 3, under Algorithm and Runtime parameters, provides specific machine limits.

The lamda safety acceleration factor determines the smoothness of running feedrate. Specifying a wrong value for lamda will definitely cause acceleration jitters or rapid fluctuations, and that will certainly end up in jerky machine feedrates. This must be avoided in all situations. Therefore, the first objective is to determine the acceptable safe value for lamda.

The strategy of finding lamda is to run the algorithm by sequentially increasing the lamda values (between 0.00 to 1.00) until rapid fluctuations or jitters in the tangential

acceleration occurs.

At the onset of tangential acceleration jitters, the safe lamda shall be the value just below it. The lower the lamda value the safer (from jitters) it shall be.

In this work, the lamda value was found at lamda = 0.18, which is considered the threshold for acceleration safety factor.

Algorithm executions were conducted on all ten(10) parametric curves to determine the most suitable single value of lamda. The values of lamda were increased sequentially in steps, from lamda equals 0.10 to 0.18, 0.20 and 0.50.

The user feedrate command for all the ten(10) curve executions was set at a common FC = 20 mm/s. The results in the ongoing pages describe the determination of lamda.

The idea of a single lamda for the algorithm is such that, this lamda shall be safe for running all curves. Otherwise, every time a user wants to run a different parametric curve, the user will have to determine lamda specifically for the curve.

First, the view of acceleration jitters shall be presented so that the user is able to recognize acceleration jitters. A jerk is a sudden rapid spike (high increase) in acceleration, while a jitter is a cyclic low value fluctuation.

One example of acceleration jitters is shown Figure [4.8] on the next page, where the algorithm was executed on the Snailshell curve with Lamda = 0.50 at FC 20. Another example on acceleration jitters for the Ribbon-100L curve running Lamda = 0.50 and FC = 20, is shown in Figure [4.10]. The graphs are presented in landscape mode for clarity.

4.3.5 Examples of acceleration jitters (landscape)

For the first example Snailshell Tangential Acceleration Jitters in Figure [4.8], the figure is divided in two parts. The upper plot is for parameter range ($u = 0.00$ to $u = 0.50$), while the lower plot is for ($u = 0.50$ to $u = 1.00$).

In the next figure on the Snailshell Close View Tangential Acceleration Jitters, Figure [4.9], the upper plot is the expanded parameter u -range ($u = 0.30$ to $u = 0.40$) while the lower plot is for ($u = 0.70$ to $u = 0.80$). The clearly visible "comb-like" fluctuations are definitely tangential acceleration jitters.

For the second example Ribbon-100L Tangential Acceleration Jitters shown in Figure [4.10], severe jitters were discovered for the algorithm execution. The large black bands for jitters are obviously visible. Similarly, the expanded view in the next Figure [4.11] confirms jitters upon close inspection.

In both of the cases above, choosing $\lambda = 0.50$ for the acceleration safety factor is definitely not acceptable because of jitters. It must be lower.

As a side note, it is important to realize that there is a large number of interpolated points between a single interval like ($u = 0.30$ to $u = 0.40$). For the illustrative Teardrop curve, there are about 600 points in the range ($u = 0.30$ to $u = 0.40$) because the Total Interpolated Points (TIP) calculated by the algorithm for the Teardrop curve is about 6000 points.

The large number of interpolated points plotted in each $\Delta u = 0.10$ throughout the parameter range ($u = 0.00$ to $u = 1.00$) makes the plot credible. The discussion on Total Interpolated Points (TIP) for all ten(10) curves is provided in a later section.

Figure 4.8: Example Snailshell Tangential Acceleration Jitters

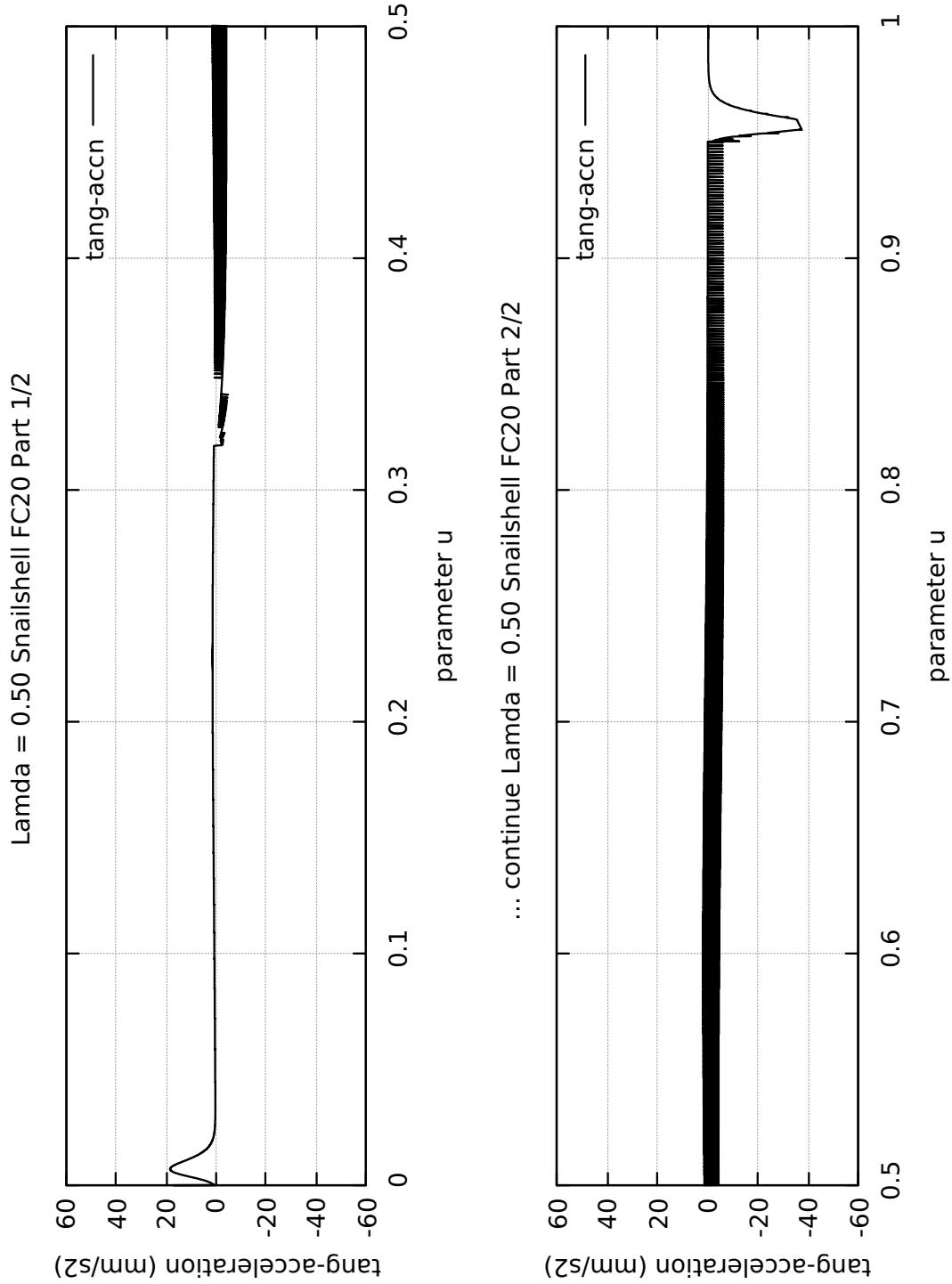


Figure 4.9: Example Snailshell Close View Tangential Acceleration Jitters

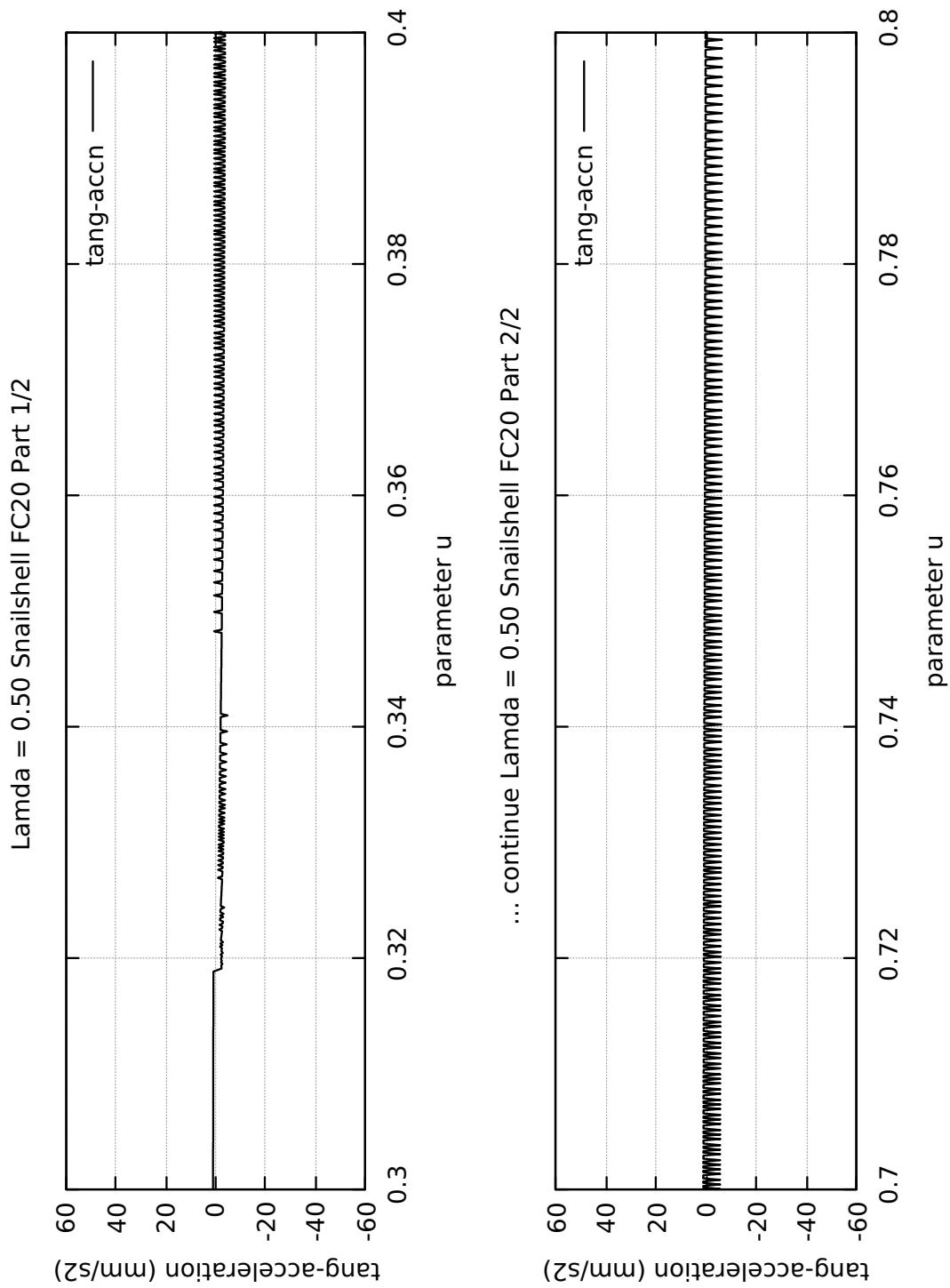


Figure 4.10: Example Ribbon-100L Tangential Acceleration Jitters

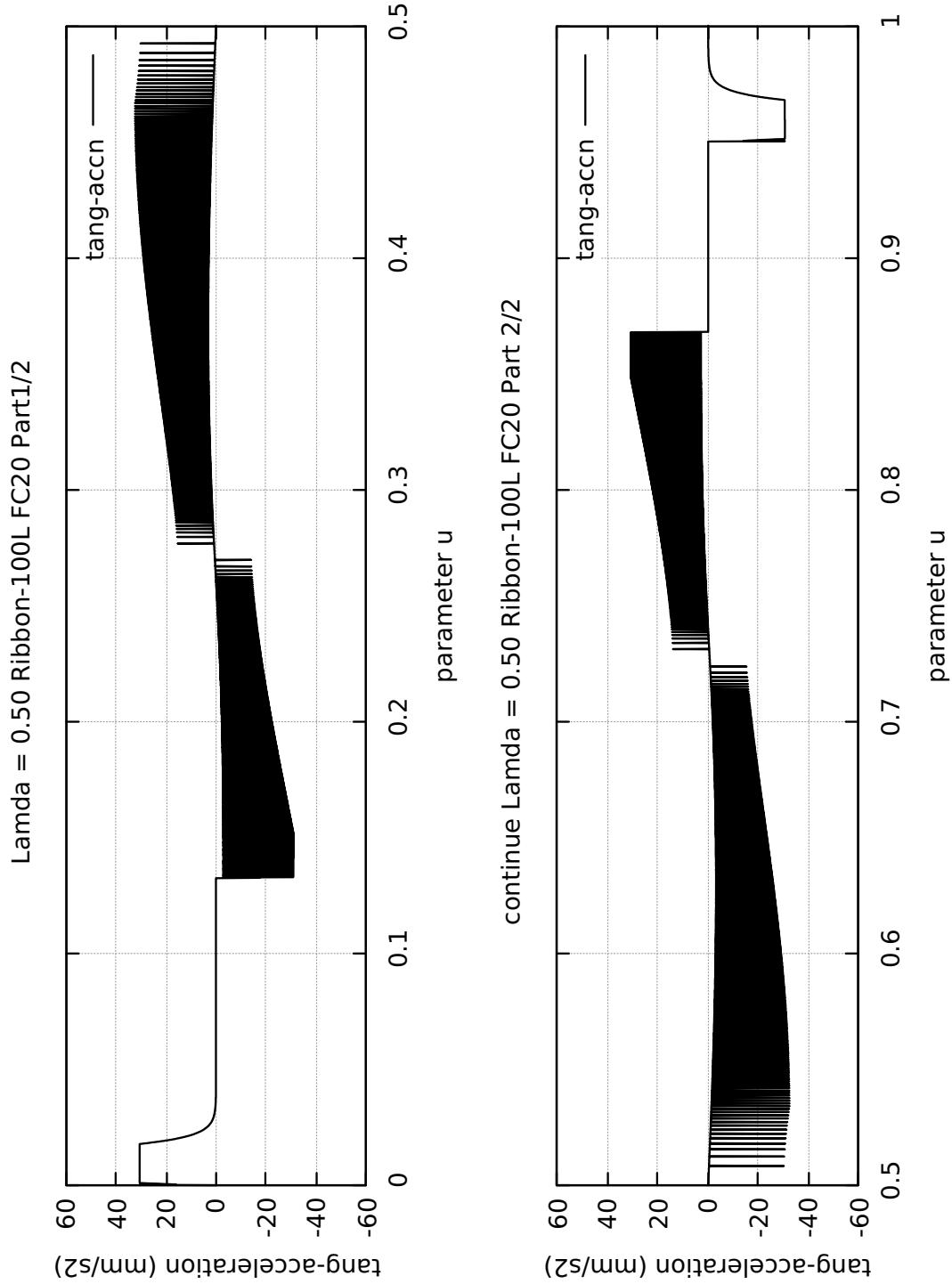
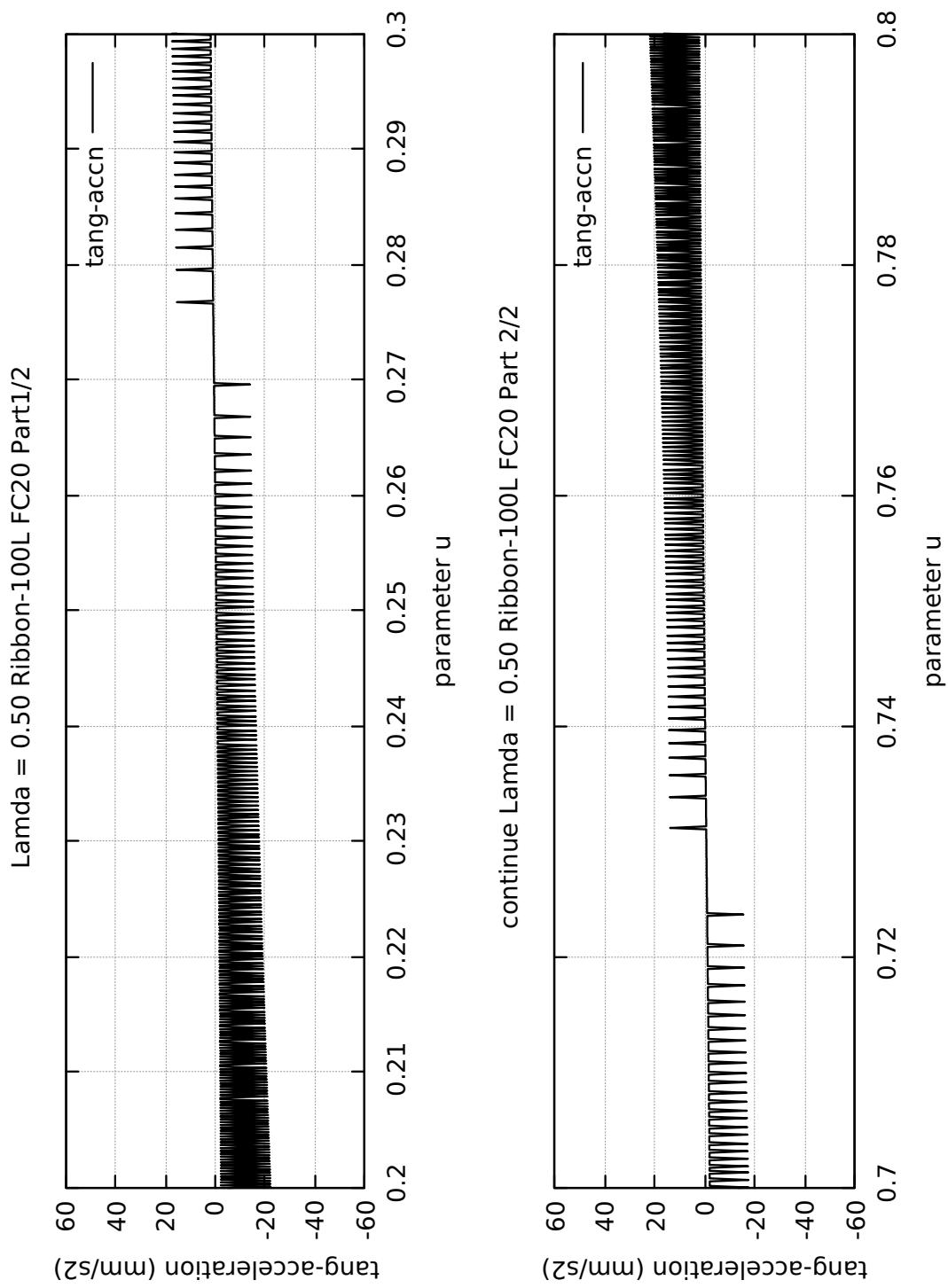


Figure 4.11: Example Ribbon-100L Close View Tangential Acceleration Jitters



4.3.6 Results for finding acceptable lamda

The next ten(10) pages of results contain the tangential acceleration profiles for all ten(10) parametric curves handled in this work. Each page contains four(4) plot executions for one particular curve at lamda = 0.10, 0.18, 0.20 and 0.50. All executions run at Feedrate Command FC = 20 mm/s.

On each page, lamda = 0.10 for the upper left plot, lamda = 0.18 for the lower left plot, lamda = 0.20 for the upper right plot, and lamda = 0.50 for the lower right plot. Attention must be made to lamda = 0.18 that is located at the lower left plot on the page.

The list of parametric curves with tangential acceleration profiles are as follows.

Table 4.1: Results for finding acceptable lamda

No.	Curve Type	Reference Link	lamda = 0.18	Remarks
1	Circle	Figure [4.12]	good	
2	Ellipse	Figure [4.13]	good	
3	Teardrop	Figure [4.14]	good	lamda = 0.20 is bad
4	Butterfly	Figure [4.15]	suspect	To recheck
5	Snailshell	Figure [4.16]	good	
6	Skewed-Astroid	Figure [4.17]	good	
7	Ribbon-10L	Figure [4.18]	good	
8	Ribbon-100L	Figure [4.19]	good	
9	AstEpi	Figure [4.20]	good	
10	SnaHyp	Figure [4.21]	suspect	To recheck

For those curves that the lamda = 0.18 is suspect, a close inspection for jitters is required.

All plots in the next pages are in landscape mode.

4.3.7 Rechecking acceptable lamda

Close inspection for the following two(2) curves both at lamda = 0.18, Butterfly and SnaHyp revealed the following.

Table 4.2: Results for rechecking acceptable lamda

No.	Curve Type	Reference Link	lamda = 0.18	Remarks
4A	Butterfly	Figure [4.22]	good	2X magnification
4B	Butterfly	Figure [4.23]	good	Close up view
10A	SnaHyp	Figure [4.24]	good	2X magnification
10B	SnaHyp	Figure [4.25]	good	Close up view

In conclusion, lamda = 0.18 is the acceptable acceleration safety factor to be applied in the execution of all ten(10) parametric curves.

Figure 4.12: Circle Lamda Safety Factor Threshold

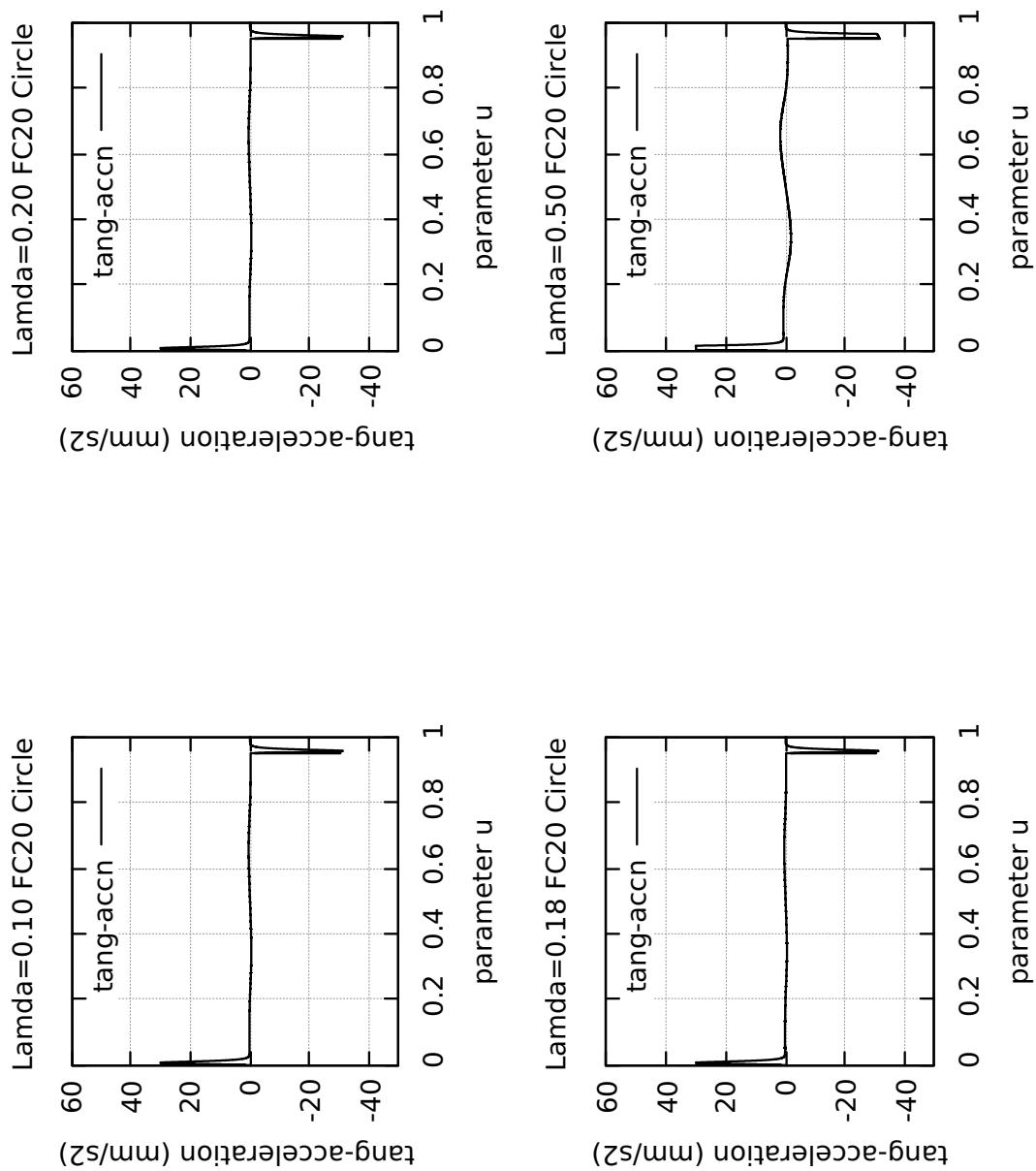


Figure 4.13: Ellipse Lamda Safety Factor Threshold

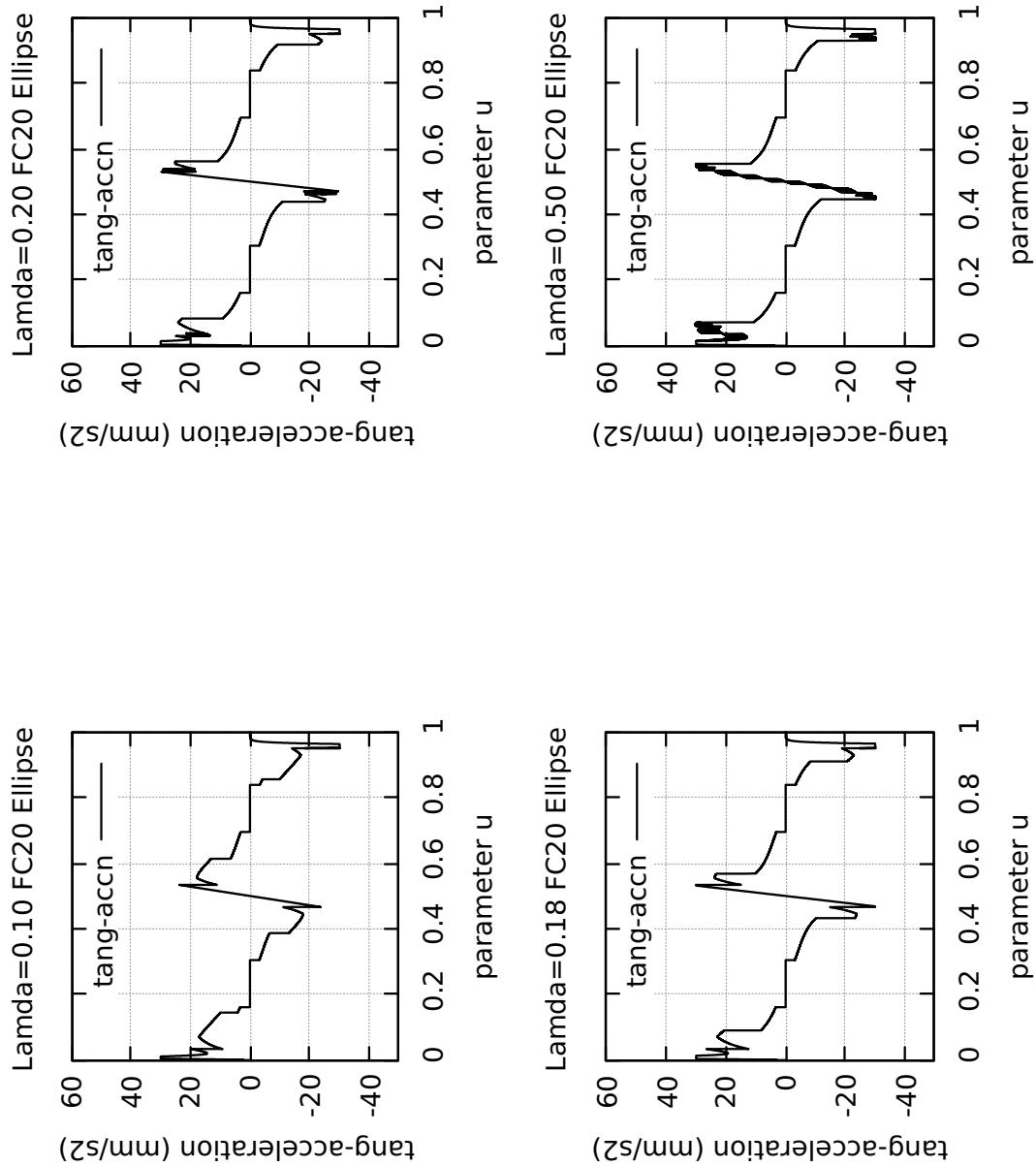


Figure 4.14: Teardrop Lamda Safety Factor Threshold

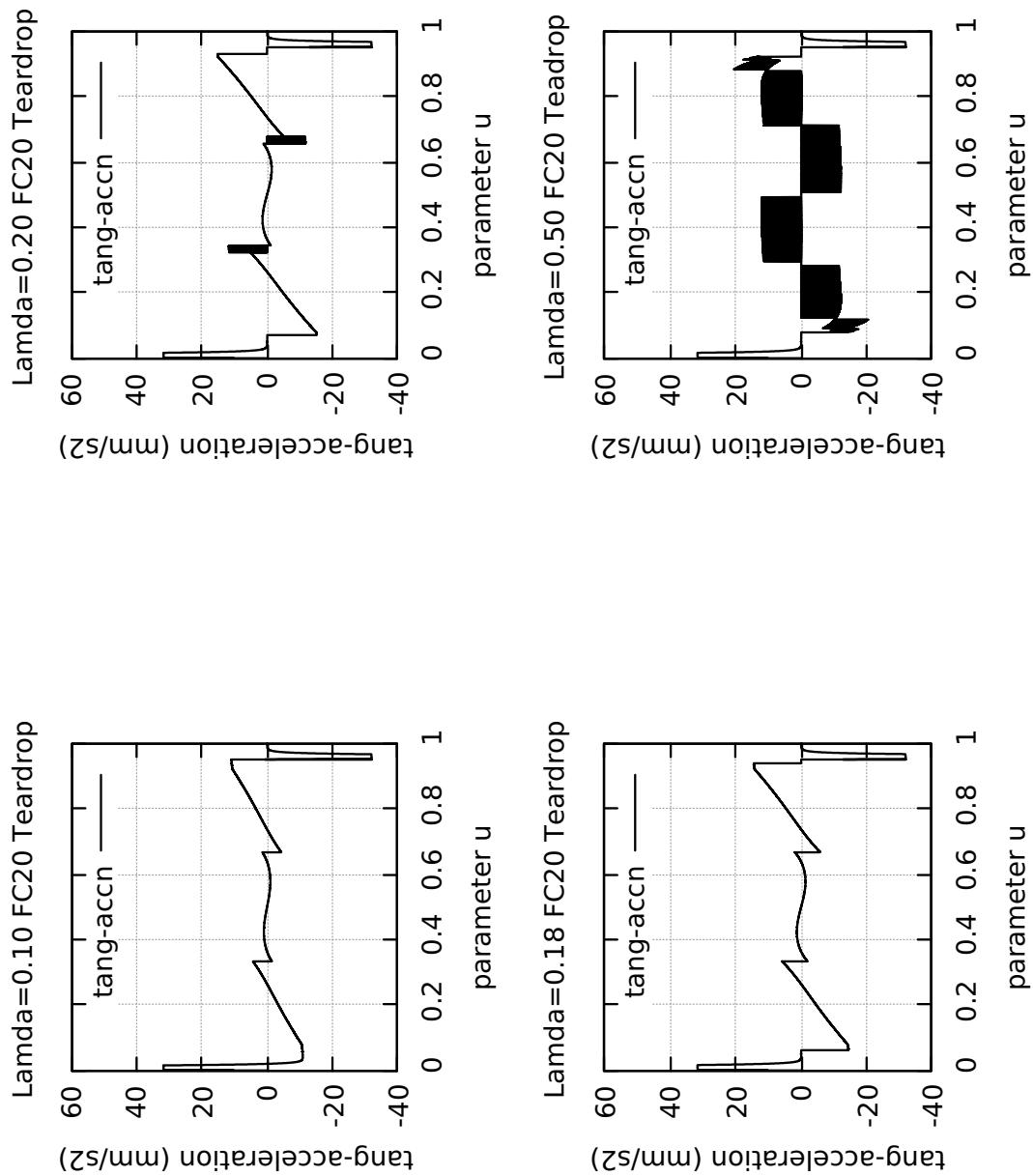


Figure 4.15: Butterfly Lamda Safety Factor Threshold

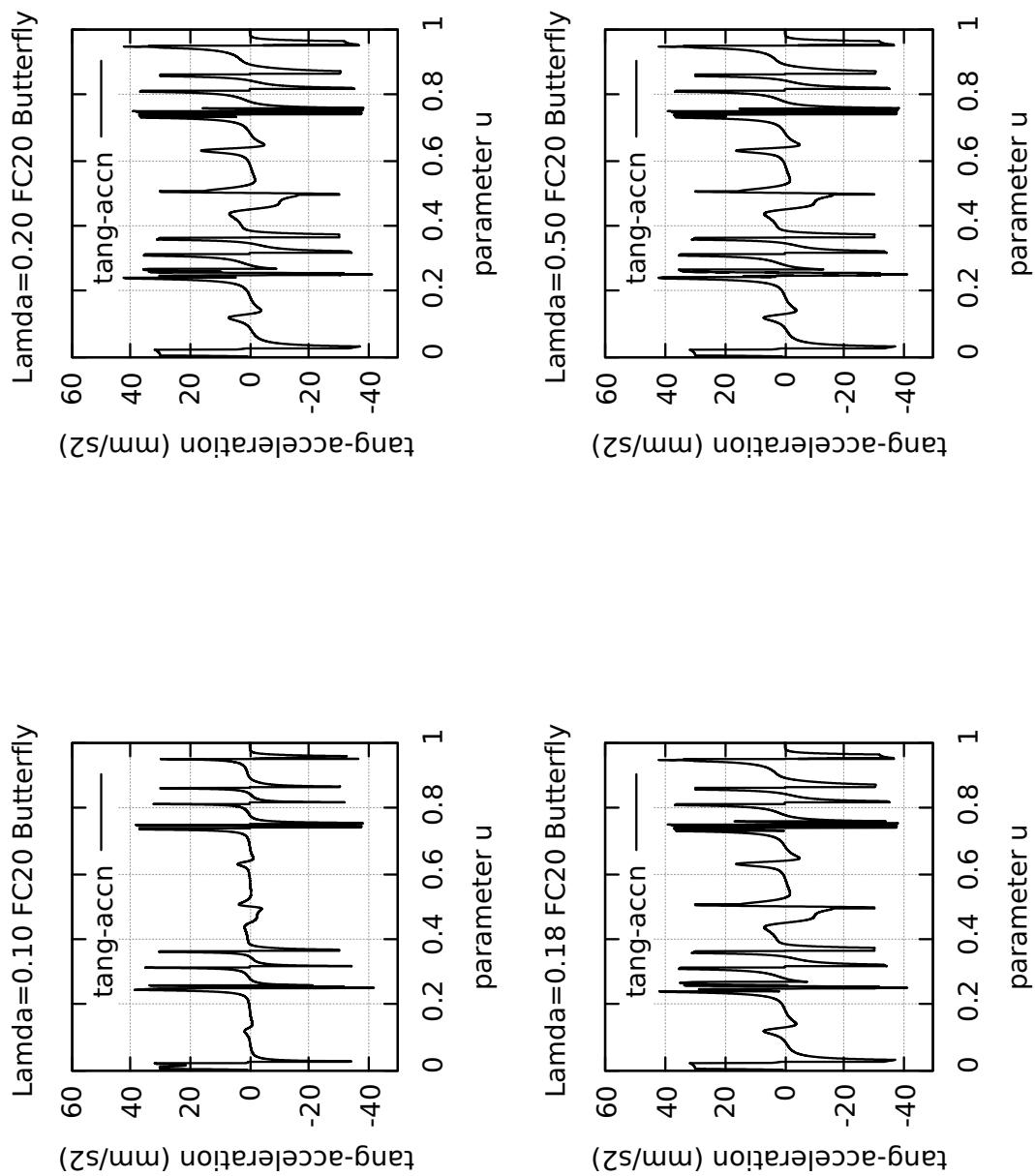


Figure 4.16: Snailshell Lamda Safety Factor Threshold

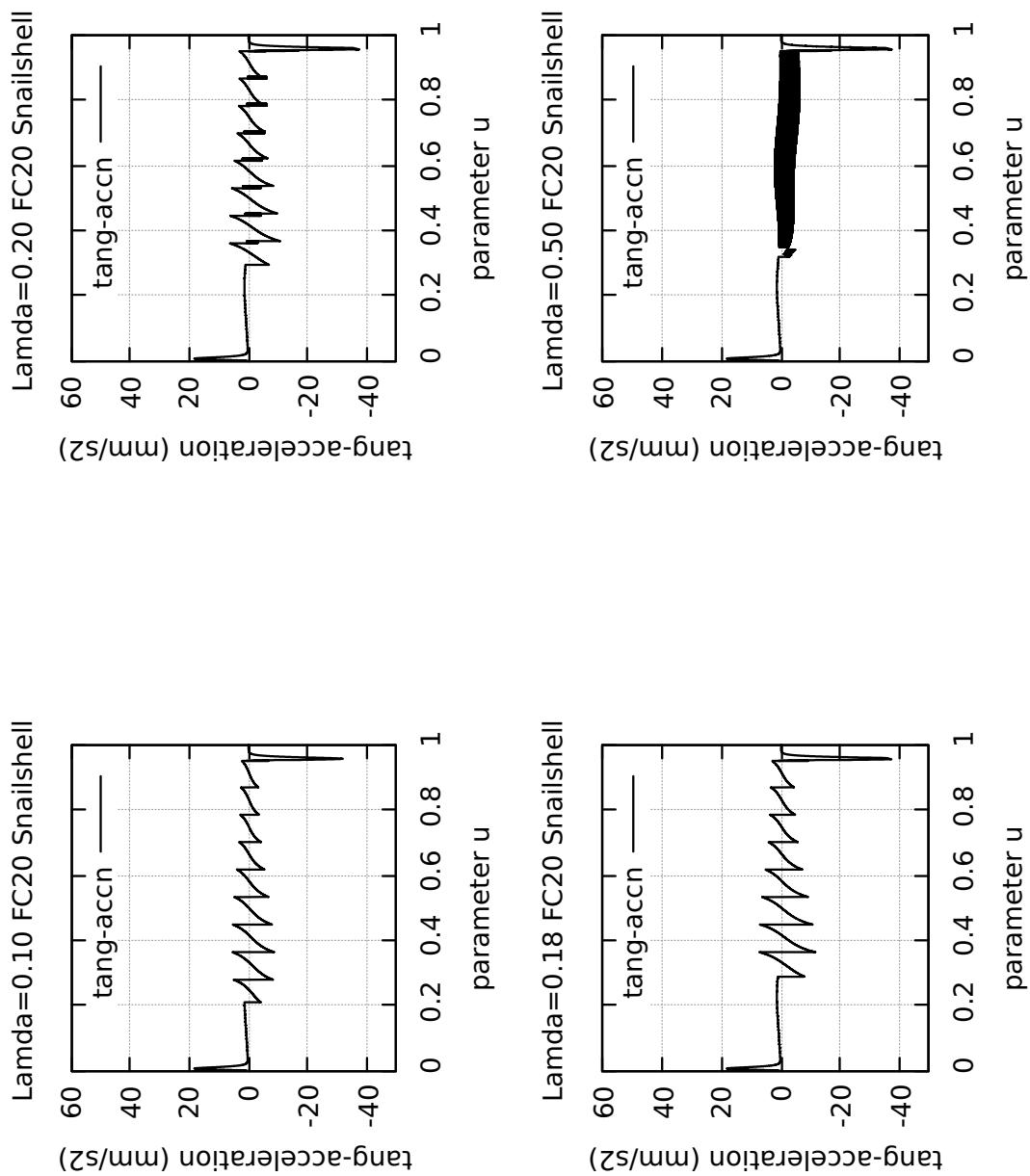


Figure 4.17: Skewed-Astroid Safety Factor Threshold

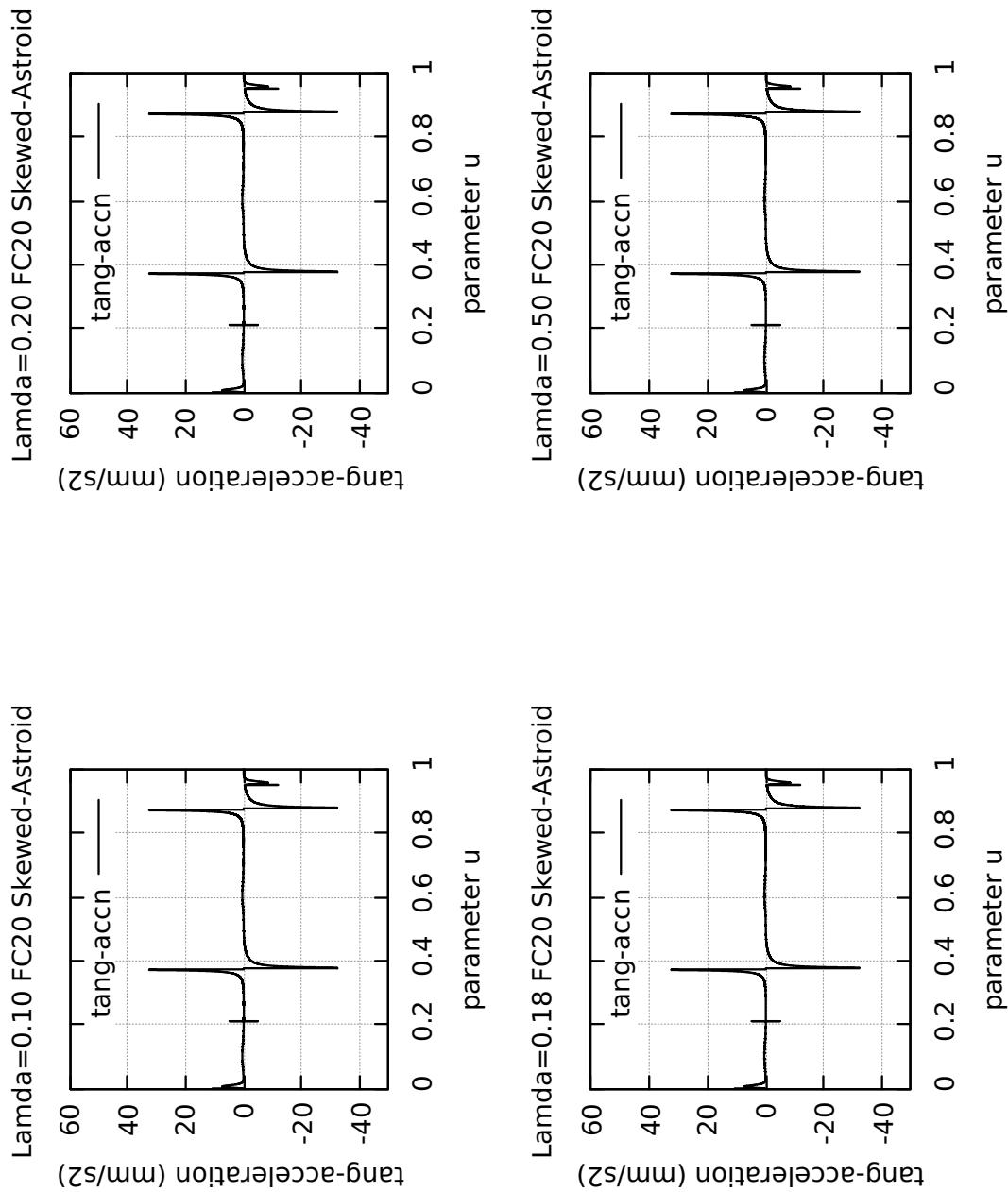


Figure 4.18: Ribbon-10L Lamda Safety Factor Threshold

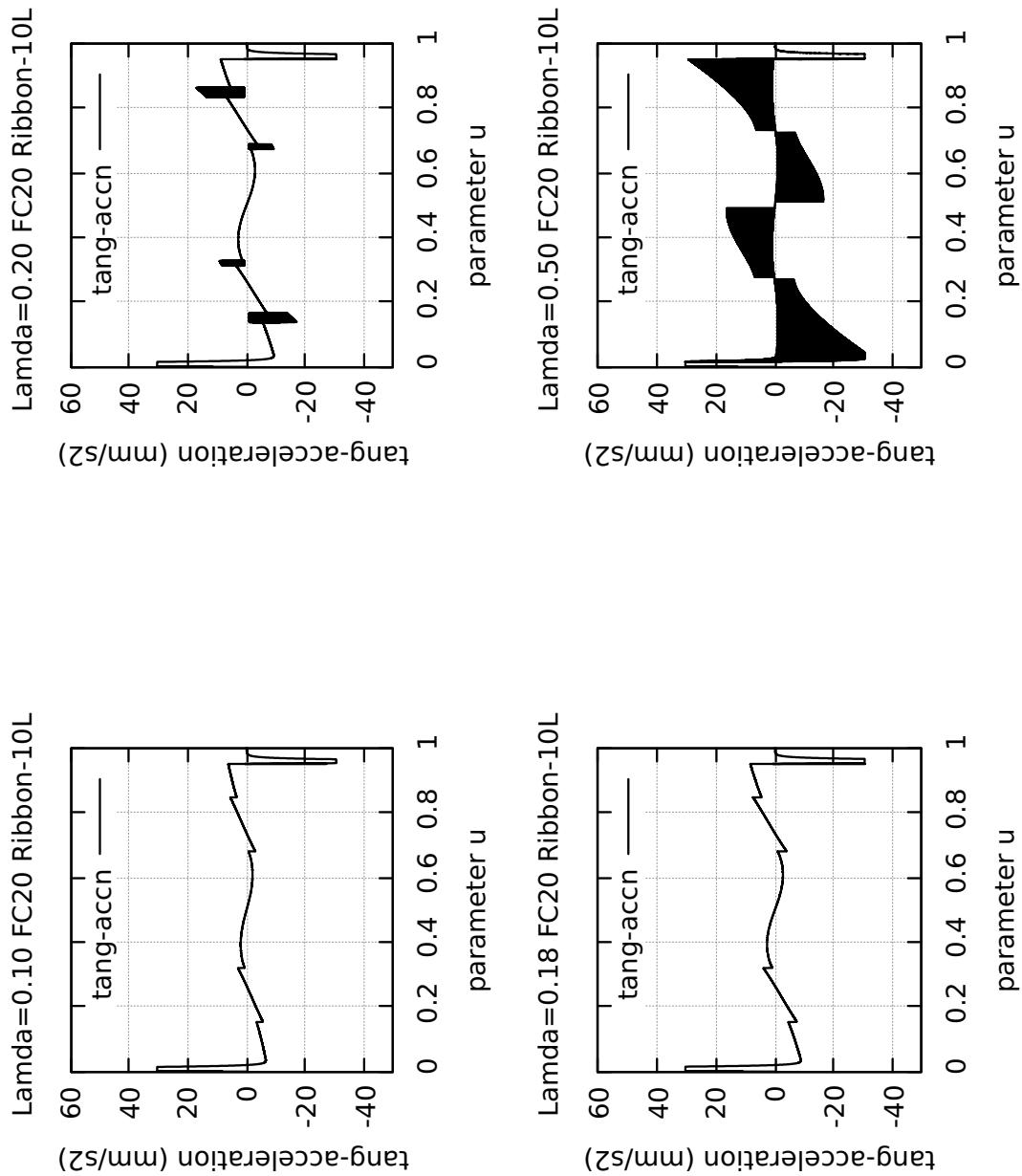


Figure 4.19: Ribbon-100L Lamda Safety Factor Threshold

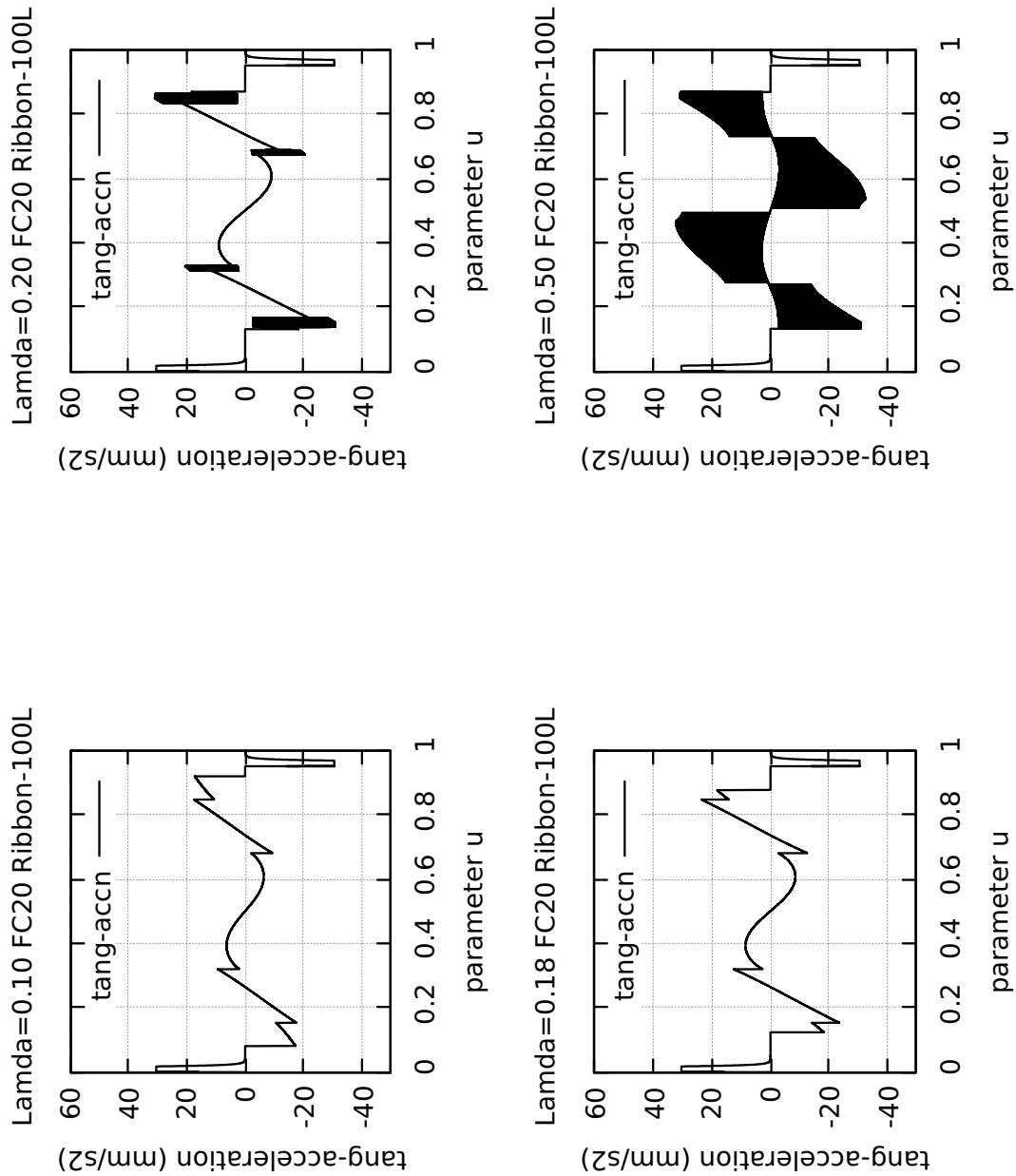


Figure 4.20: AstEpi Lambda Safety Factor Threshold

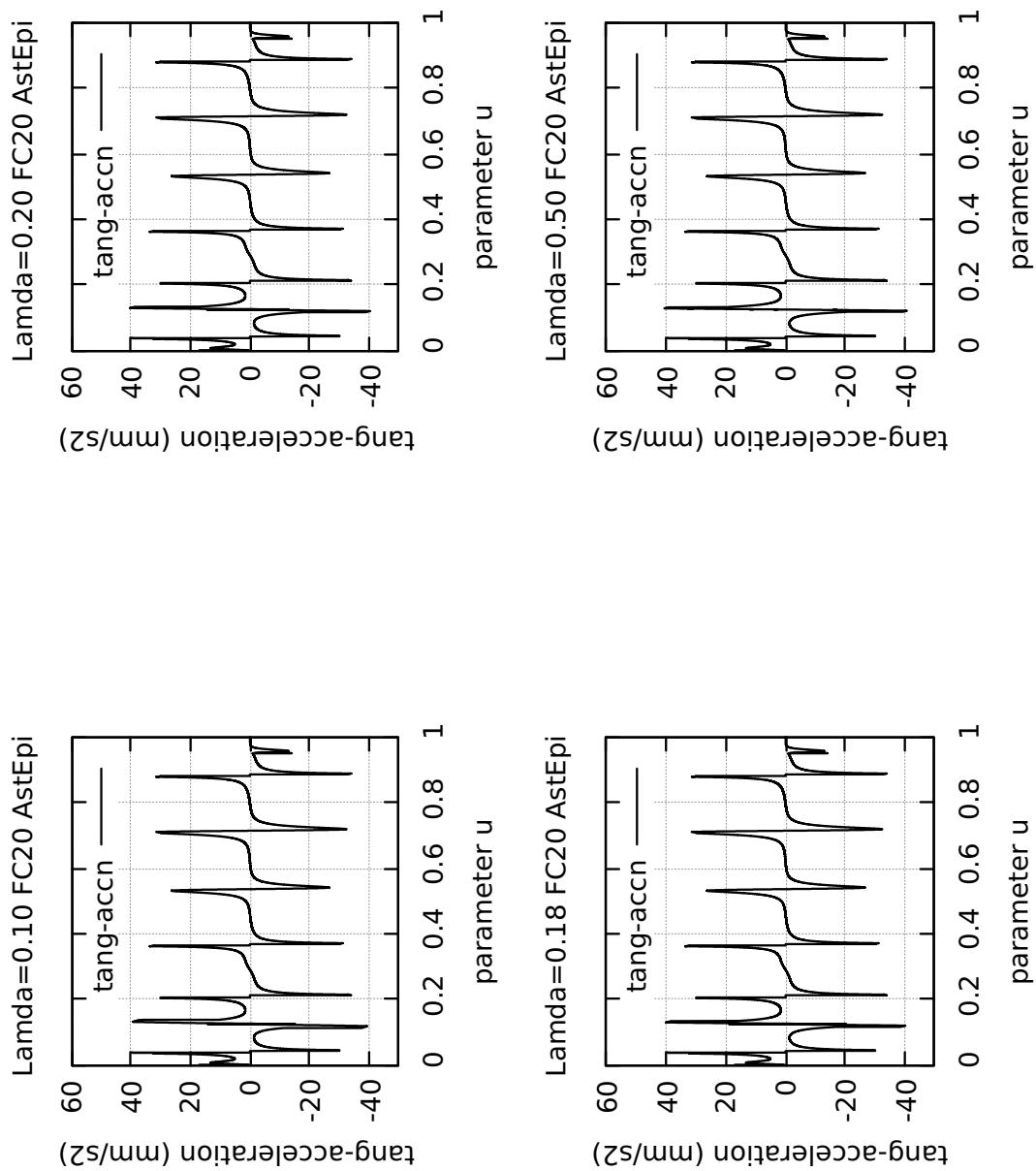


Figure 4.21: SnaHyp Lamda Safety Factor Threshold

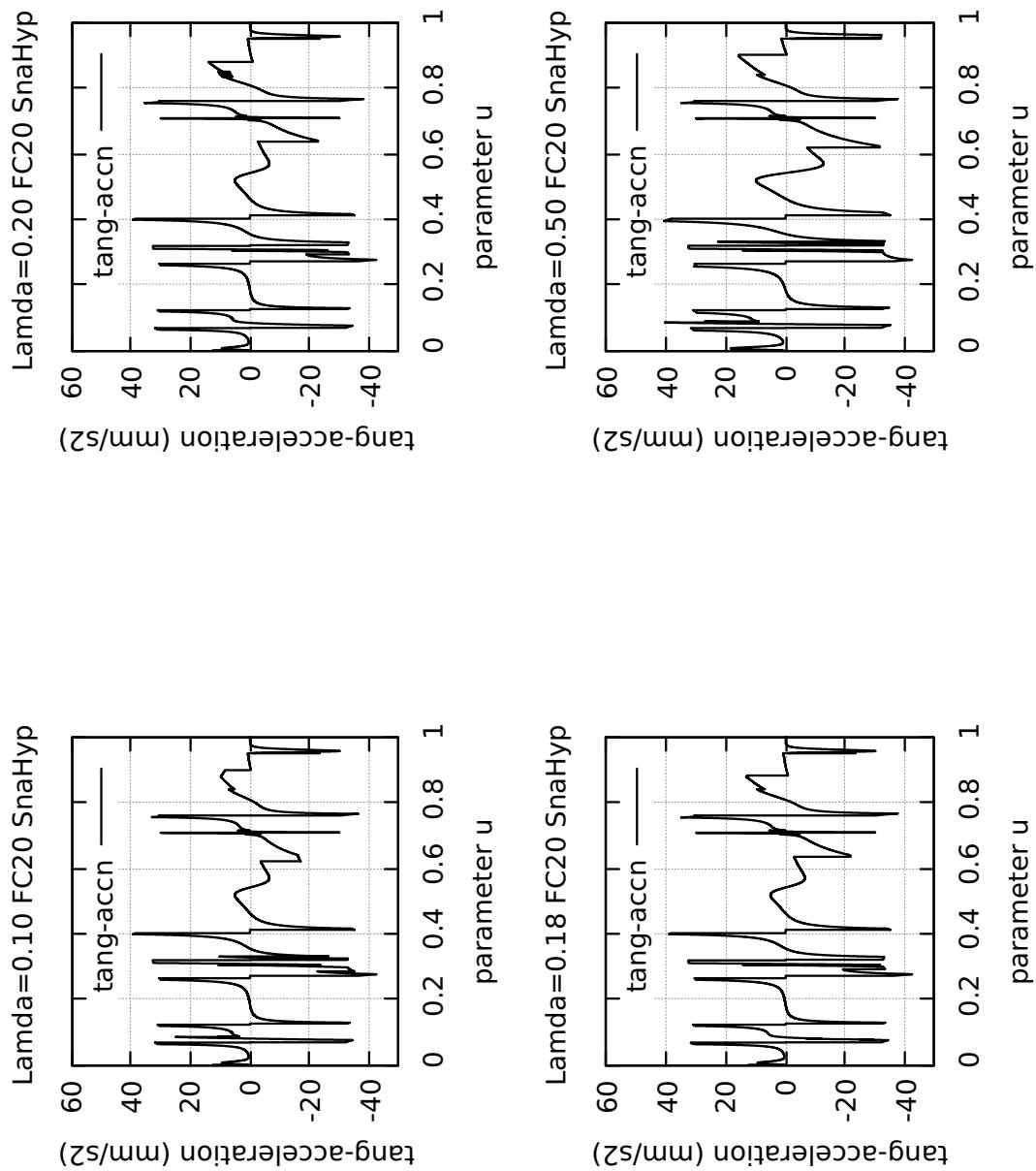


Figure 4.22: Butterfly Tangential Acceleration Magnified 2X

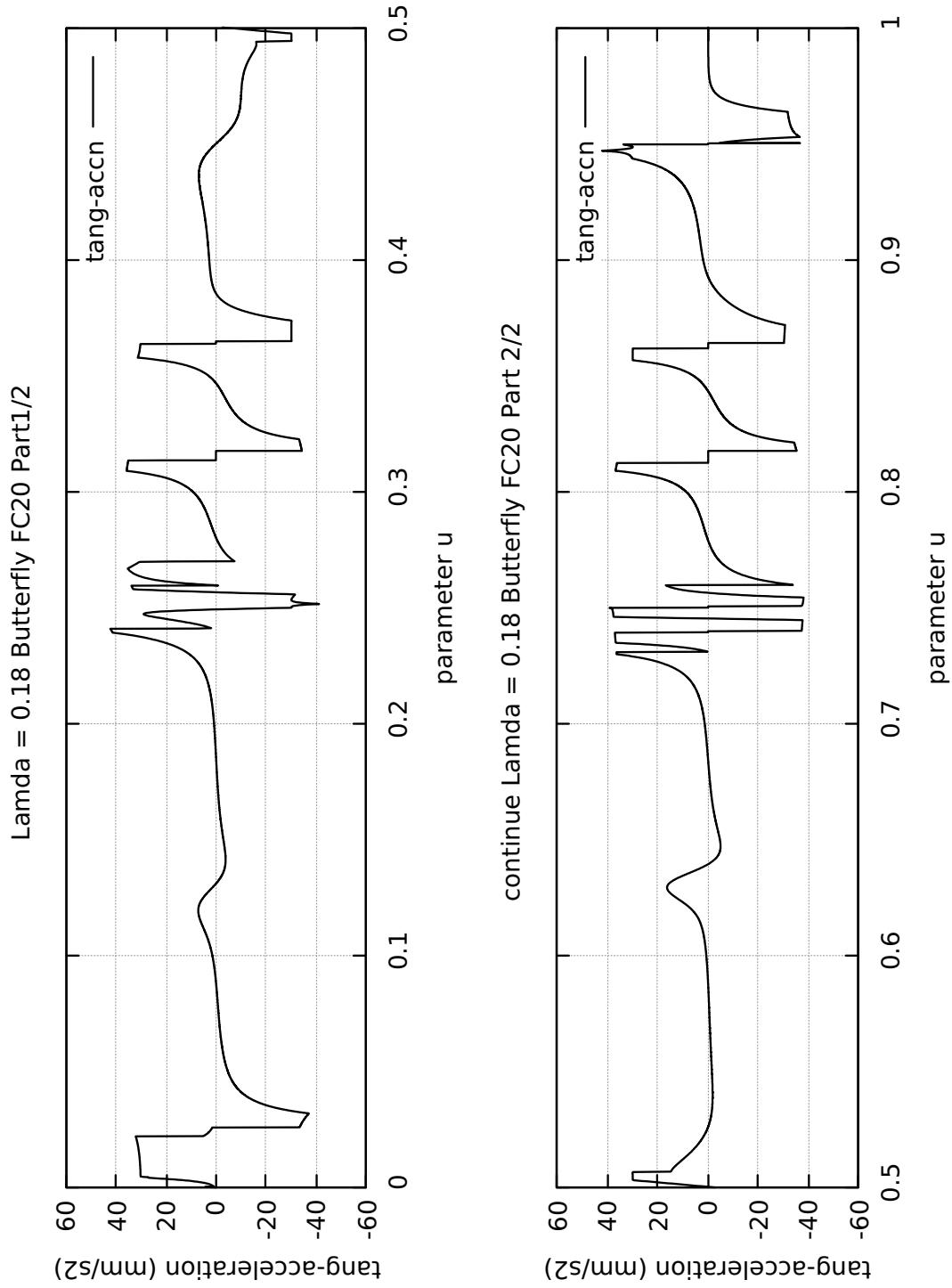


Figure 4.23: Butterfly Tangential Acceleration Closeup View

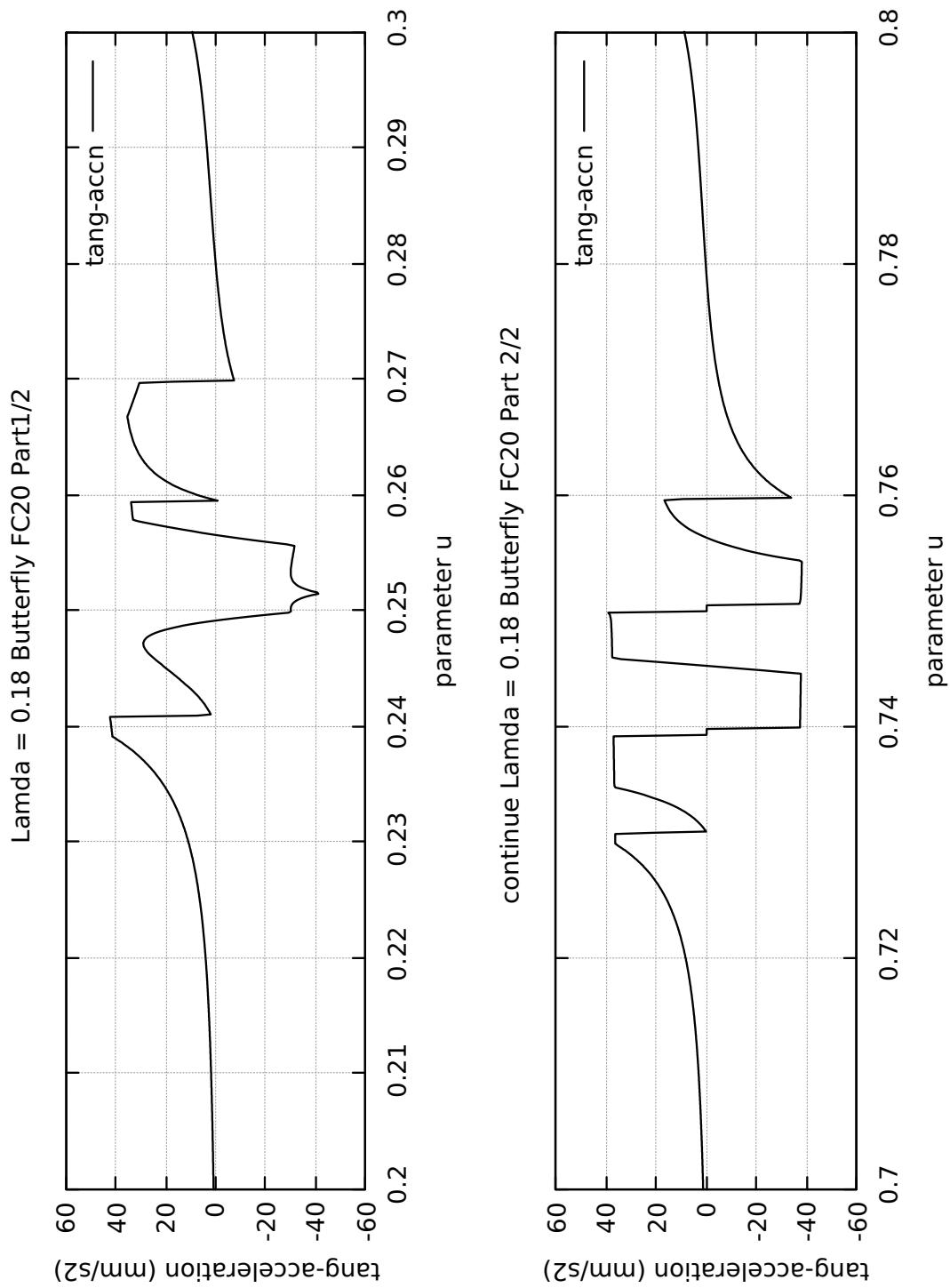


Figure 4.24: SnaHyp Tangential Acceleration Magnified 2X

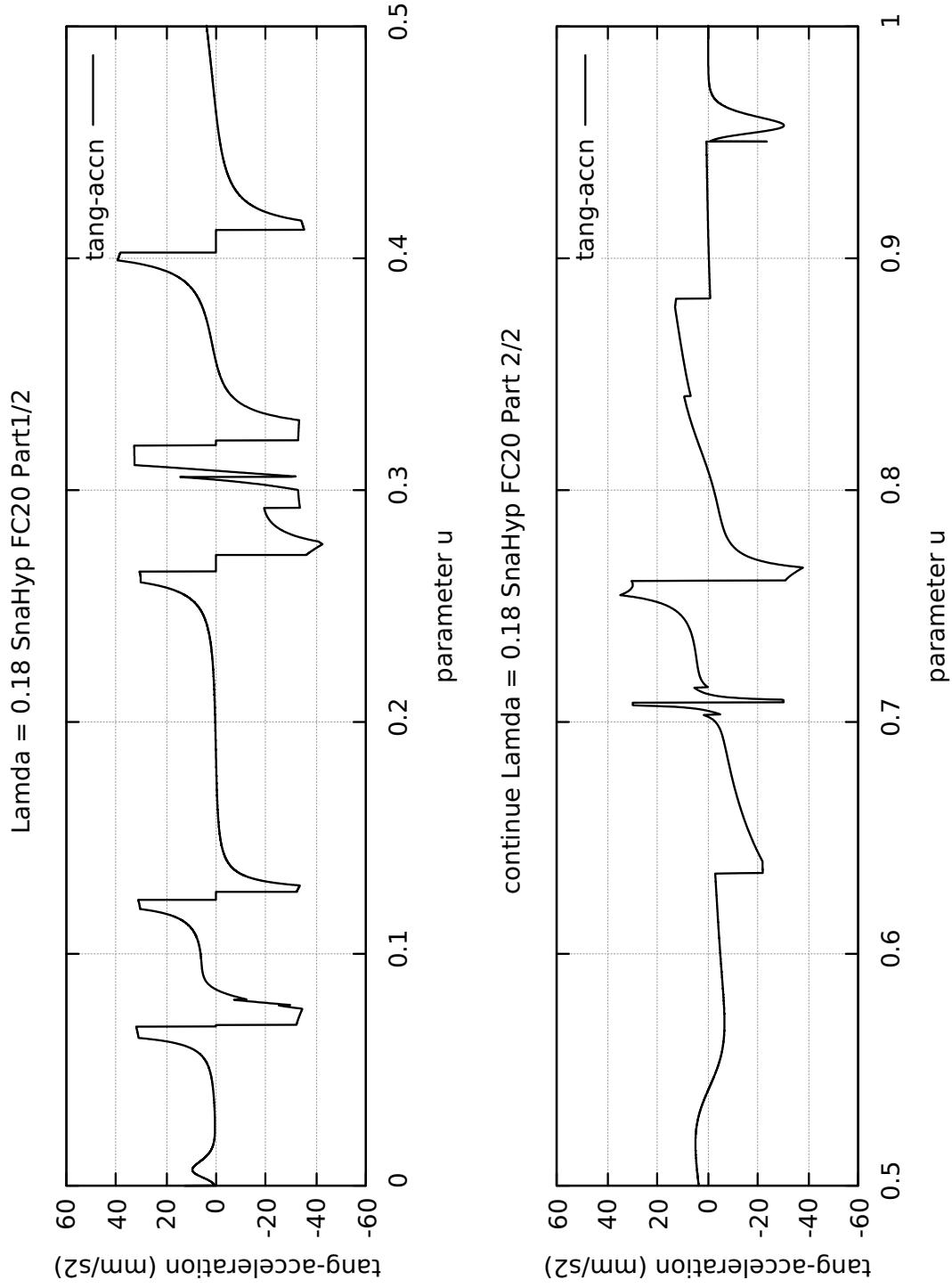
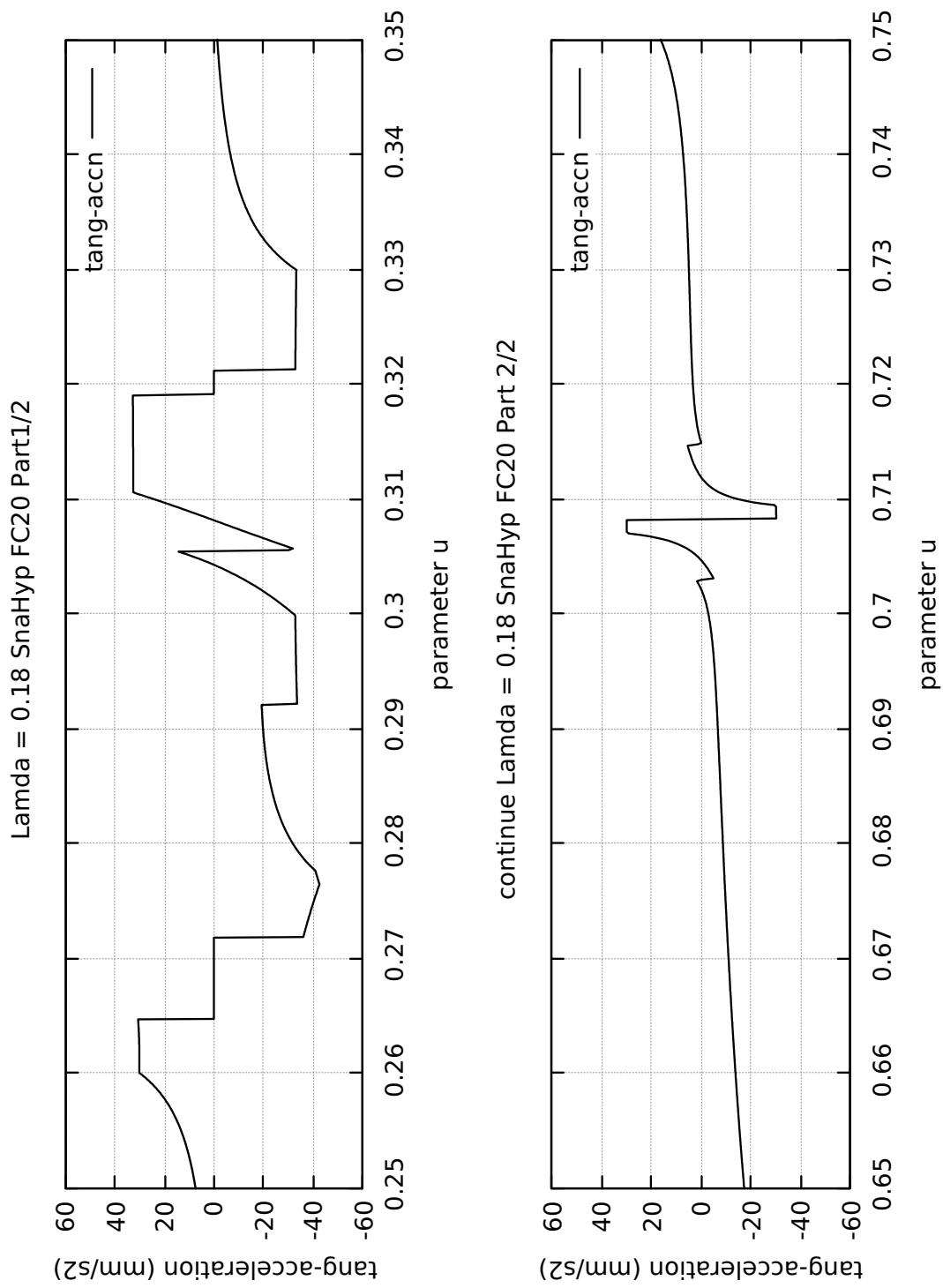


Figure 4.25: SnaHyp Tangential Acceleration Closeup View



4.4 TEARDROP CURVE FOR ILLUSTRATION

4.4.1 Teardrop parametric curve

Instead of displaying the execution results of all ten(10) parametric curves in this chapter, only the full results of the Teardrop curve shall be displayed for illustration. The results for the rest of the parametric curves are provided in individual appendices specific to the corresponding curve.

The Teardrop curve was used as an illustrative example due to its special characteristics. It has the combination of all curve features studied in this work. The Teardrop curve contains, for example, gentle convex turns, near straight line sections, a sharp pointed apex akin to a cusp at the origin, a closed loop curve, symmetrical about the y-axis reflection but non-symmetrical about the x-axis reflection, and overall curve is not origin centered.

4.4.2 Rest of other nine(9) parametric curves

The rest of the other curves studied in this work are essentially extreme variations of the Teardrop curve.

1. Circle in Appendix Reference [App.3].

The Circle curve was selected primarily to prove that the algorithm is performing correctly. For the Circle, the calculation of circumference by the standard geometry formula ($C = 2 * PI * Radius$) can be validated and verified by comparing to the results of the algorithm for the total sum-arc-length and the total sum-chord-length, respectively. Since the circle spans a central angle of (2*PI) radians or 360 degrees, the results for the total sum-arc-theta calculated by the algorithm is another worthy comparison. To add challenge the interpolation algorithm, the radius of the circle was chosen to be 79, a prime number divisible only by one and itself. In this work, the Circle curve is therefore featurefull and not featureless.

The algorithm validation and verification for the calculation of circumference and total subtended angle (sum-arc-theta) of the Circle is provided in section reference [4.7.1], later in this chapter.

2. Ellipse in Appendix Reference [App.4].

The Ellipse curve is basically the extreme elongation of the Circle curve. Similarly, it was selected primarily to prove that the algorithm calculation of the perimeter of the Ellipse using Ramanujan approximation formula, using a and b for the semi-major and semi-minor lengths, of the Ellipse, respectively. This is the algorithm's validation and verification against the Ellipse curve. The results for the total sum-arc-theta calculated by the algorithm is another validation and verification for the Ellipse curve. Unlike the Circle which has a constant Radius of Curvature rho (its own radius), the Ellipse curve as expected have variations in its Radius of Curvature which must be symmetrical.

The algorithm validation and verification for the Ellipse is provided in section reference [4.7.2], later in this chapter.

3. Teardrop in Appendix Reference [App.5].

The Teardrop curve can be viewed as a perfect circle or an ellipse curve compressed at one end to become a tip apex. The Teardrop curve is the base curve used for illustration purposes. The characteristics of the Teardrop curve have been mentioned in the previous section. The discussions for the rest of the curves will follow in a similar manner to the discussions of the Teardrop curve.

4. Butterfly in Appendix Reference [App.6].

The Butterfly curve can be viewed as many Teardrop curves (6 lobes) of various sizes joined together at a central point, the origin ($x = 0.00$, $y = 0.00$). However, the start and ending points in the Butterfly curve for parameter u travel, that is,

$u = 0.00$ and $u = 1.00$, respectively, are not at the origin, even though the curve is origin-centered. The Butterfly curve in fact, starts and ends at ($x = 7.182818$, $y = 7.182818$), respectively. This is a challenge to the algorithm. In addition, the Butterfly curve has the most complicated direction of travel, and radius of curvature rho(u) profiles.

5. Snailshell in Appendix Reference [App.7].

The Snailshell curve is an "off-grid" curve. It was selected because the Radius of Curvature rho(u) decreases continuously as the curve spirals toward the center. It was also chosen because it is an open curve, unlike the Circle, Ellipse and Teardrop which are closed curves. This Snailshell curve challenges the interpolation algorithm for its continuously and cyclically decreasing curvature, and its open nature.

6. Skewed-Astroid in Appendix Reference [App.8].

The Skewed-Astroid curve is essentially a square diamond figure, compressed at its linear edges to become four(4) concave curves (not convex). The Skewed-Astroid curve is also elongated on the y-axis to make its two vertical apexes very sharp cusps, compared to the two x-axis apexes which are broader. The cusps and the elongation of its concave sides are challenges to the interpolation algorithm.

7. Ribbon-10L in Appendix Reference [App.9].

The Ribbon-10L curve is essentially another Teardrop curve with two(2) extended almost linear legs supporting the apex. It is an off centered figure (not centered at the origin) and sized extremely small (4 mm by 4 mm), which is about the surface area of a finger tip. This is a true challenge to the interpolation algorithm for its scale down features.

8. Ribbon-100L in Appendix Reference [App.10].

The Ribbon-100L curve is just a ten-times scale up of the Ribbon-10L curve (40 mm by 40 mm). This is another challenge to the interpolation algorithm for its scale up features. It is interesting to see the comparative results of both the Ribbon-10L and Ribbon-100L curves generated by the interpolation algorithm.

9. AstEpi in Appendix Reference [App.11].

The AstEpi curve is another off-grid curve. It is basically a mathematical linear combination of the standard Astroid and standard Epicycloid curve equations. The result is the AstEpi curve having three closed-loops that is origin-centered, and symmetrical about the 45 degree line between the x and y axis. The challenge for the algorithm is in handling the combination of mathematical functions.

10. SnaHyp in Appendix Reference [App.12].

The off-grid SnaHyp curve is another mathematical linear combination of the standard Snailshell and standard Hypotrocoid curve equations. Instead of being in closed-loop form like the AstEpi curve, this SnaHyp curve is open-looped and looks like a random continuous curve. The challenge to the algorithm is the handling of arbitrary, open-loop and random curve segments.

From the above discussions, it should be realized that there is a clear and specific purpose in the selection of each of the ten(10) parametric curves used in this work. It is worthy to mention that the realtime interpolation algorithm developed in this work passed all of the diverse challenges mentioned above.

4.5 RESULTS OF TEARDROP CURVE

4.5.1 Plot of Teardrop curve

The characteristics of the Teardrop curve is shown in Figure [4.26] on the next page. The curve starts from the origin ($x = 0.0, y = 0.0$) where $u = 0.0$ and ends again at the origin ($x = 0.0, y = 0.0$) where $u = 1.00$. The movement direction is counter-clockwise.

The Teardrop curve contains, for example, gentle convex turns, near straight line sections, a sharp pointed apex akin to a cusp at the origin, a closed loop curve, symmetric about the y-axis reflection but non-symmetric about the x-axis reflection, and overall curve is not origin centered.

4.5.2 Teardrop Direction of Travel

The direction of travel for the Teardrop curve is shown in a 3D plot in Figure [4.27] on the next page. Since the direction is determined by the increasing (decreasing) value of parameter u , it can be seen through the z-axis the locations of the (x,y) points on the base ($x-y$ plane) as parameter u moves from ($u = 0.00$) to ($u = 1.00$).

The 3D direction curve is simple for the Teardrop curve, but can be complicated for example: the Butterfly, Skewed-Astroid, AstEpi and SnaHyp curves.

Figure 4.26: Plot of Teardrop curve

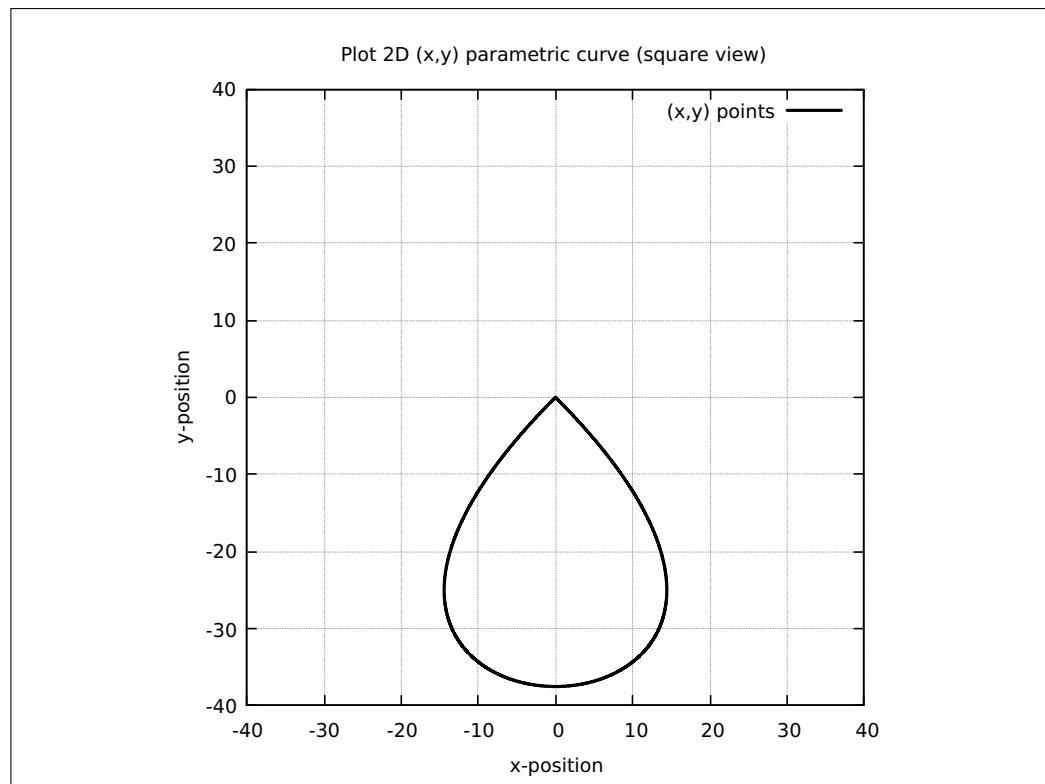


Figure 4.27: Teardrop Direction of Travel 3D

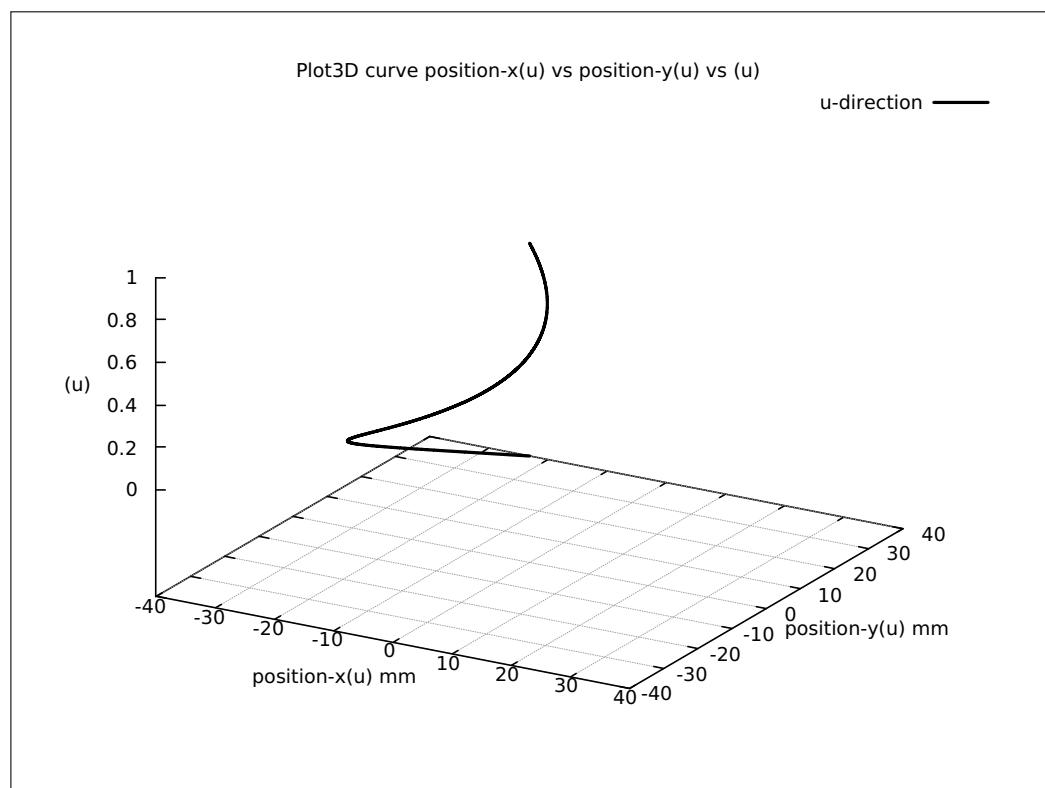
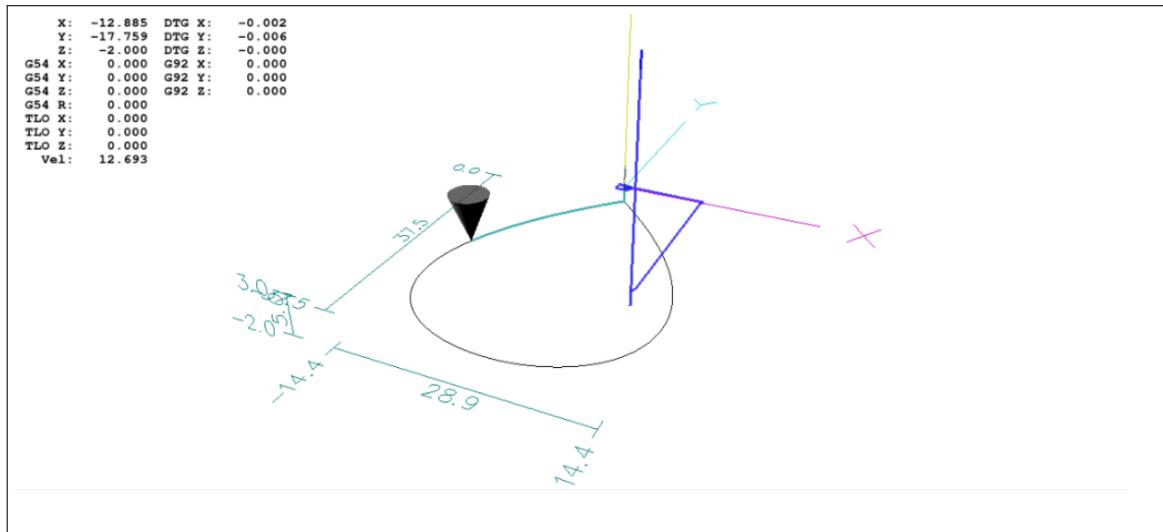


Figure 4.28: Teardrop Perspective View 3D in LinuxCNC-Axis



4.5.3 Teardrop Perspective View 3D in LinuxCNC-Axis

The perspective view of the Teardrop curve in Figure [4.28] above is a real live validation and verification of the curve running on the LinuxCNC-Axis machine. The software interface panel displays the current position of the cutting tool (cone) as the algorithm drives the tool along the full curve path.

In addition, the interface panel also displays the current (x,y) position and velocity (feedrate) of the tool. More will be mentioned on this subject in the section under LinuxCNC-Axis Simulation Run Validation [4.8.3] further in this chapter.

4.5.4 Teardrop Radius of Curvature

The Radius of Curvature rho, for the Teardrop curve is shown in Figure [4.29] on the next page. The correctness of the Radius of Curvature rho(u), is such that it must follow the corresponding curve it represents.

For the Teardrop, the value of rho(u) starts high at the beginning because it is near a straight line, and then slowly curve inward thus rho(u) becomes lower, and at the end rho(u) rises high again approaching a near straight line as it reaches its end point.

Note that the term "near straight line" have been used because a "true straight line" represents the Radius of Curvature as being infinite, and that there is "no curving" effect at all. On the other hand, a zero value for the Radius of Curvature rho(u), represents a single point, and the curving effect is meaningless. For a typical value of rho(u) like 20, it means the arc segment at that u point is curving like a perfect circle of radius 20.

Note that the Radius of Curvature rho(u) over the entire parameter range is symmetrical because the Teardrop curve for the parameter (u) movement is symmetrical. If the parameter (u) movement is not symmetrical, for example, the (x,y) point for the start ($u = 0.00$) is not at the right position, then the Radius of Curvature rho(u) over the entire parameter range is still correct but just not symmetrical. This will be the case for some of the rest of the curves in this work.

The Radius of Curvature rho(u), in general, is a good initial visualization tool to validate and indicate correctness of shapes and features of any curve. In this work, the Butterfly has the most complicated Radius of Curvature rho(u) profile. It is shown on the next page in Figure [4.30]. Further discussion on the Butterfly curve will be made in section under Notable Results Rest of Curves [4.7].

Figure 4.29: Teardrop Radius of Curvature

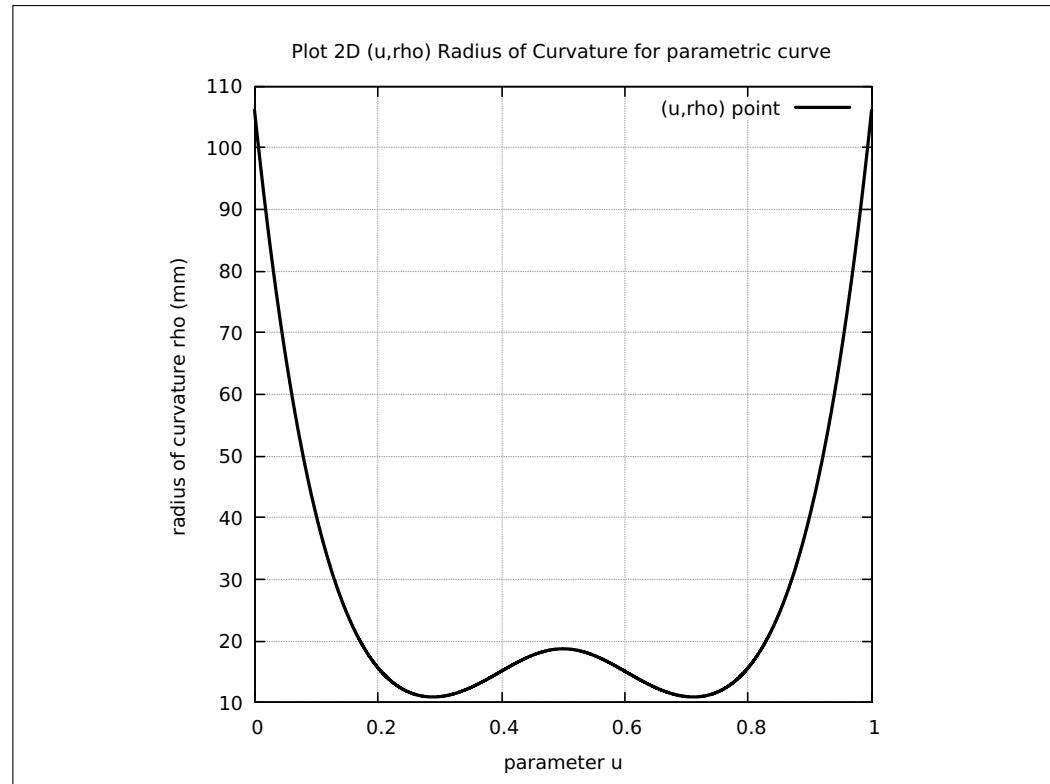
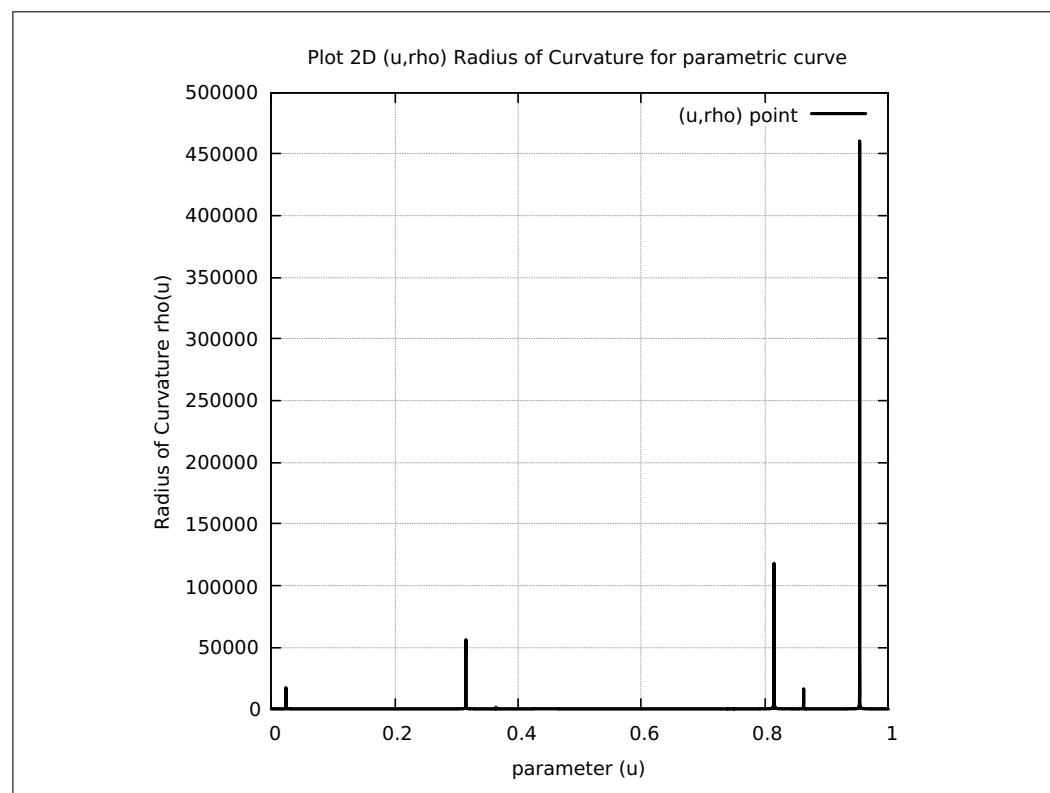


Figure 4.30: Butterfly Radius of Curvature



4.5.5 Teardrop 1st and 2nd Order Taylor's Approximation

The results of algorithm execution on the Teardrop curve for the next interpolated point $u\text{-next}(u)$ is shown in Figure [4.31] on the next page. The results comprise the first-order and second-order terms of Taylor's expansion of the curve function $C(u) = C(x(u), y(u))$, the parametric curve in 2-dimensions.

Note that two y-axes were used in this figure, one on the left in magenta and the other on the right in green. The first-order term is the magenta curve, while the second-order term is colored green. Also note that the y-scale on the left was shifted a little bit against the scale on the right. This is to facilitate the separation of the two colored curves. Otherwise, they overlap each other and the difference in values between them is not visible, that is, on the same scale. The next plot handles the difference between the two terms.

4.5.6 Teardrop (1st minus 2nd) Order Taylor's Approximation

The plot for the difference in value between the first-order and second-order terms in Taylor's approximation is provided in Figure [4.32] on the next page. Take note that in Taylor's series expansion, the largest term is the first-order term, the second-order term is smaller, and the third-order term is even smaller, so it continues to an infinite number of smaller terms, theoretically.

In addition, the sign of the successive terms in Taylor's series expansion alternates between positive (+) and negative(-), with the starting first-order term being positive (+). That is the reason the plot of the difference is calculated as (first-order - second-order) terms, thus giving a positive value. It is the same reason in separating (shifting) the two curves in the previous figure making the second order term (green) lower than the first-order term (magenta). The difference in value between the two terms exist, but is too small to be visible on the same scale in the previous figure. Therefore, the current figure displays only the difference on a single and expanded y-axis scale.

Figure 4.31: Teardrop 1st and 2nd Order Taylor's Approximation

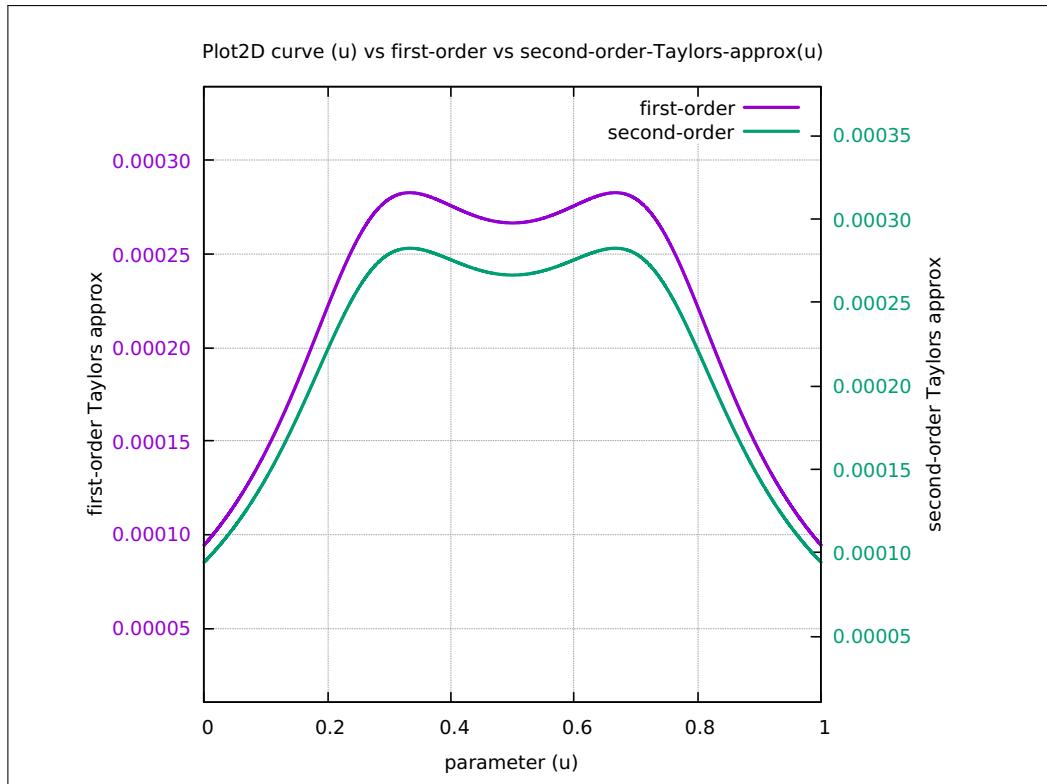
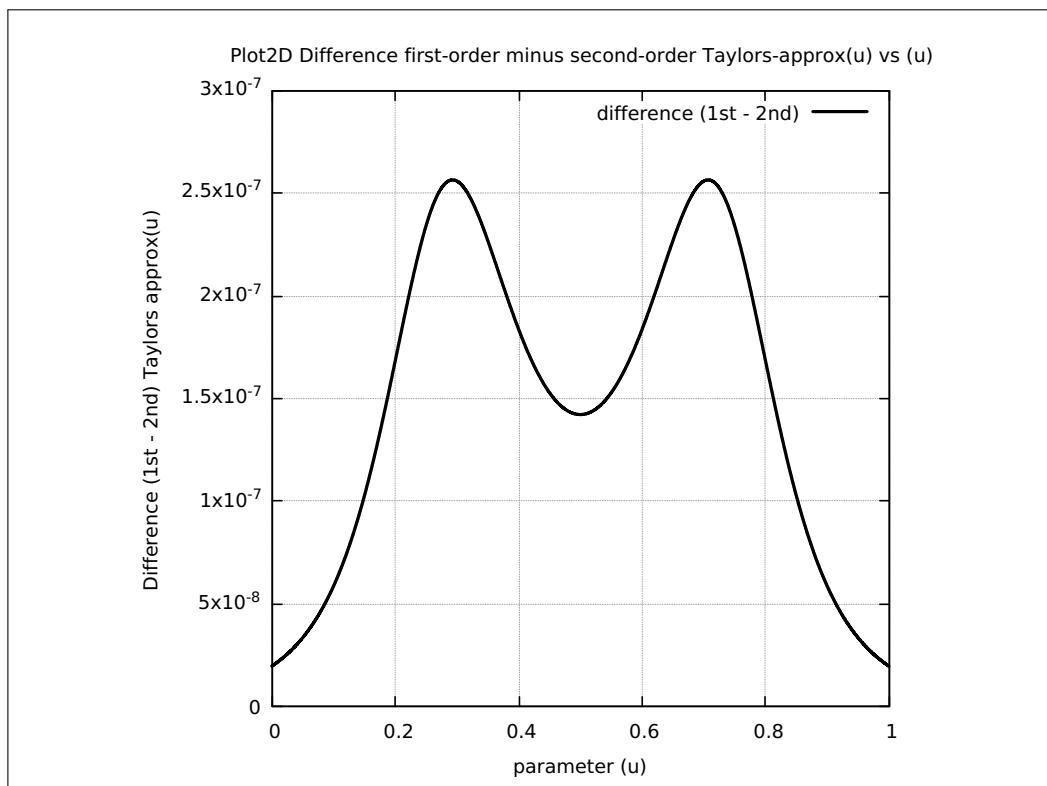


Figure 4.32: Teardrop (1st minus 2nd) Order Taylor's Approximation



4.5.7 Teardrop Chord-Error Absolute Constraint

The results for the Teardrop chord-error absolute constraint is provided in Figure [4.33] on the next page. The two colors (magenta and green) are actually the same data, that is chord-error, but are plotted on two different scales, the magenta curve using the scale on the left y-axis, while the green curve using the right y-axis.

Note that it is the same value data set but plotted on two different scales. The situation is different from the case of the first-order and second-order terms in Taylor's expansion, in which the later comprises two different calculated data sets. This means that if only the magenta curve is displayed for the chord error, a straight horizontal line will be seen in the middle section.

The idea for a second expanded scale for the green curve (same chord-error data set), is to visualize the fluctuations or jitters calculated by the algorithm. This graphical plot conclusively shows that the chord-error(u) at every u -point in the full range of ($0.00 \leq u \leq 1.00$) lies below the chord-error tolerance, which was set at 1E-6 mm.

This absolute constraint on chord-error is one of the main objectives of the algorithm in this thesis.

4.5.8 Teardrop Four(4) Feedrate Limit Components Profile

The results for the Teardrop four(4) feedrate limit components is provided in Figure [4.34] on the next page. The value of the feedrate limit at any point u , Feedrate_Limit(u), is the minimum value among the four components, evaluated at the same point u .

The figure shows the yellow curve is the most dominant (lowest value) among the 4 limit components throughout the middle range, outside of the rising S-curve and falling S-curve sections. The yellow curve is Limit-4, the Acceleration Confinement limit.

This plot is based on the Teardrop curve running at FC20 and lamda = 0.18 (restrictive acceleration confinement). If the value of lamda is increased, for example, to lamda = 0.25 (relaxed acceleration confinement), then the yellow curve will increase, thus, criss-crossing with the blue curve (Limit-3, Chord-error accuracy). This resulted in the minimum value changing between these two curves. This is how the value of the Feedrate_Limit(u) is calculated by the algorithm.

As explained in the earlier section under Results for finding acceptable lamda at reference [4.3.6], the value of Lamda = 0.25 for the Teardrop curve running at FC20 will definitely cause acceleration jitters, thus making the running feedrate jerky in the affected regions. This result violates the requirement for absolute smoothness in feedrate, which is one of the main objectives of the interpolation algorithm in this thesis.

Figure 4.33: Teardrop Chord-Error Absolute Constraint

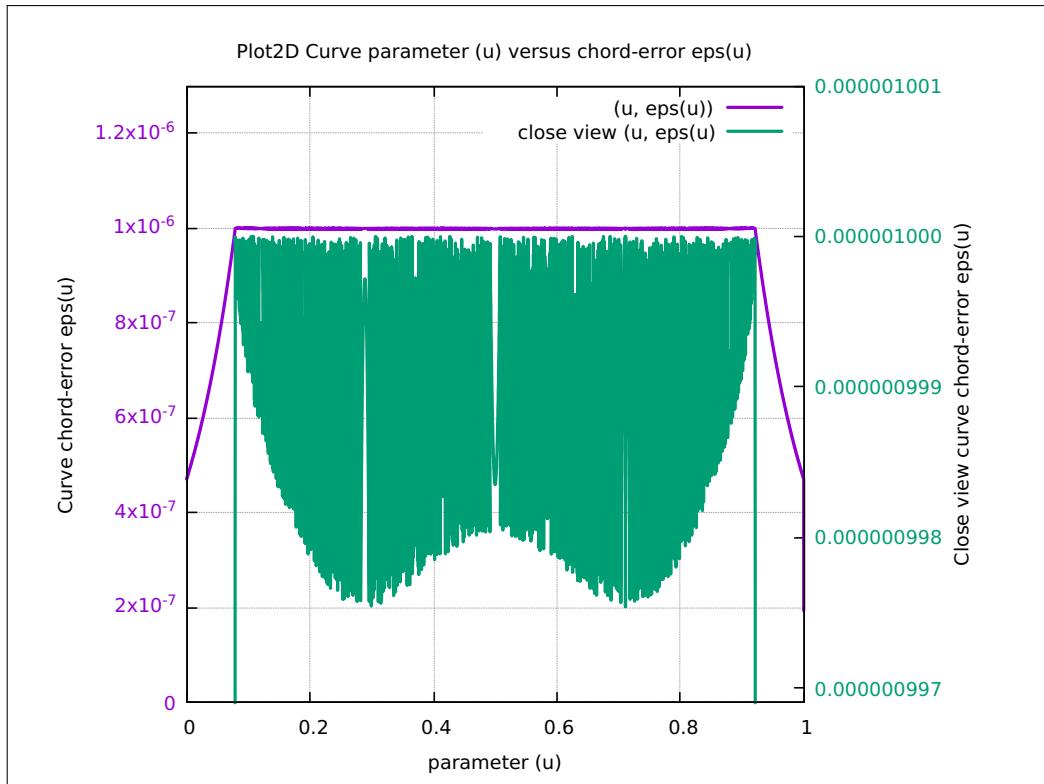
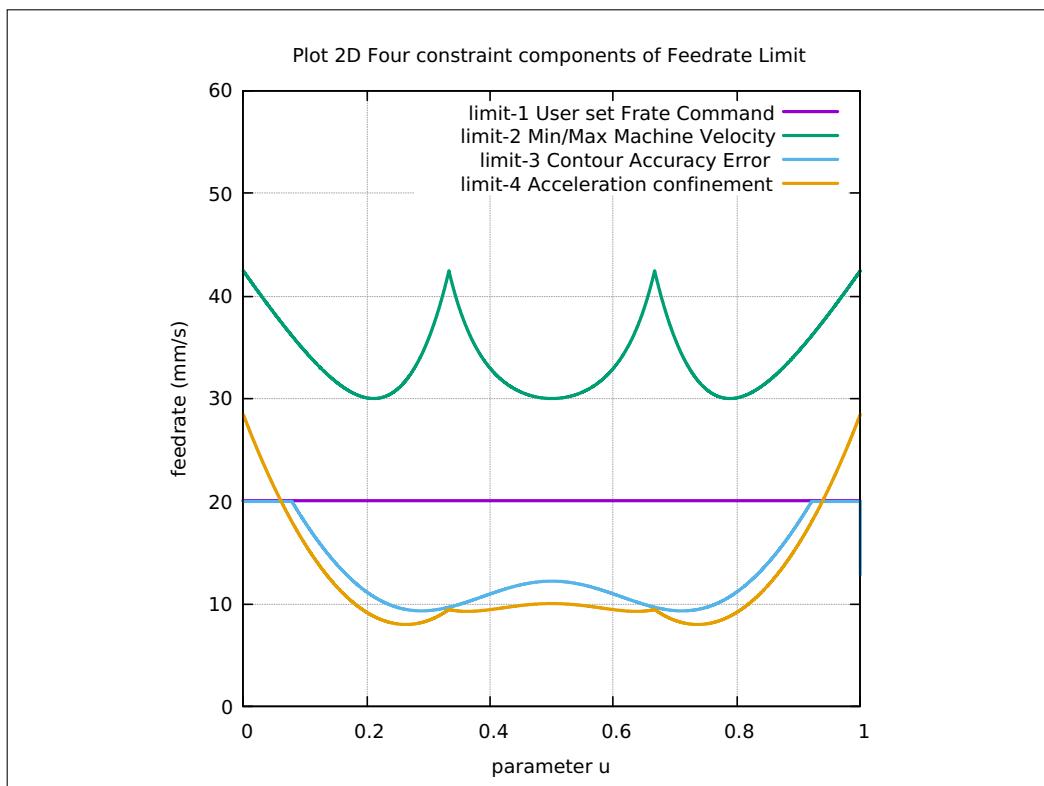


Figure 4.34: Teardrop Four Feedrate Limit Components Profile



4.5.9 Teardrop FRate Command, FRate Limit, Current FRate

The Teardrop curve execution giving results for the Feedrate Limit and the Current Feedrate is shown in Figure [4.35] on the next page. The terms current feedrate and running feedrate are used interchangeably in this document.

The Feedrate Command FC is a user specified constant value, and in this case FC = 20 mm/s. The Feedrate Limit is calculated by the interpolation algorithm at every parameter point u based on the minimum of four(4) feedrate component limits. The Current or Running Feedrate is recursively calculated by the algorithm as close to but below the Feedrate Limit.

In the figure, only two(2) of the three(3) curves are visible. This is due to the Current or Running Feedrate is seen as overlapping with the Feedrate Limit. In reality their values are close but different, and will be shown in the next figure.

4.5.10 Teardrop Current Feedrate Absolute Constraint

Figure [4.36] displays the difference in value between the Feedrate Limit and current or running feedrate for the Teardrop curve.

The display shows positive or zero value difference in blue color for the entire range of u values of the Teardrop curve, without exception. This means that the feedrate limit is always higher than the running feedrate, even though the difference is very small. There are no negative differences. Therefore, the running feedrate is constrained to be below the feedrate limit.

This absolute constraint on running feedrate is one of the main objectives of the algorithm in this thesis.

Figure 4.35: Teardrop FRate Command, FRate Limit and Current FRate

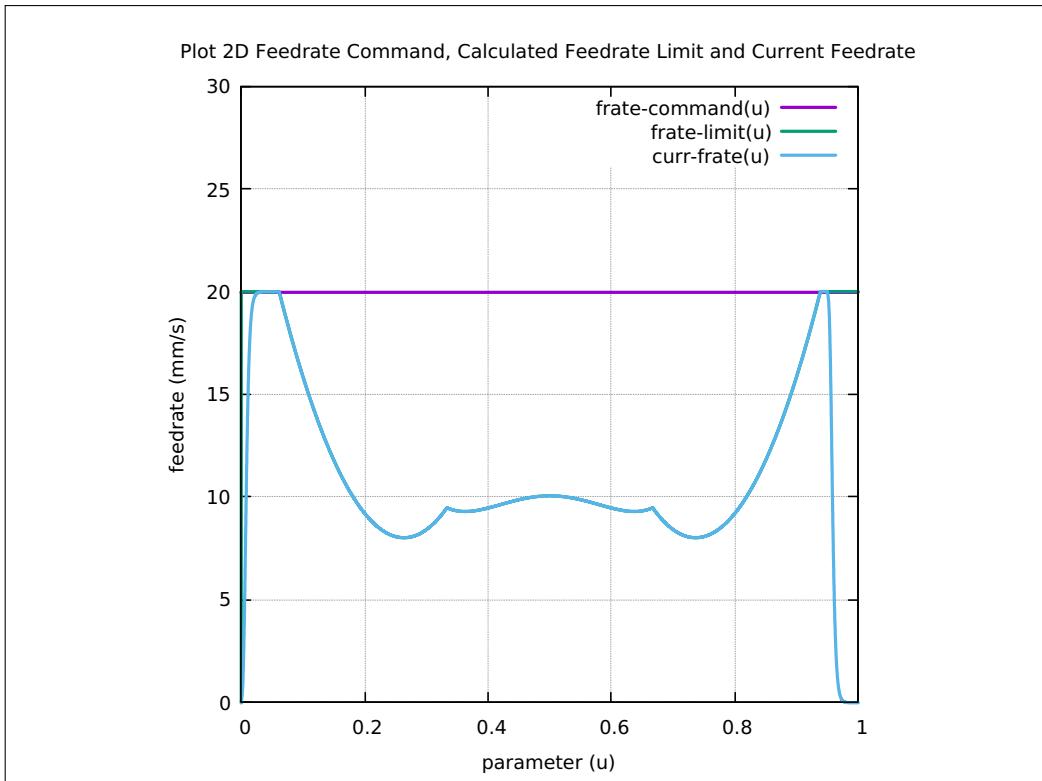
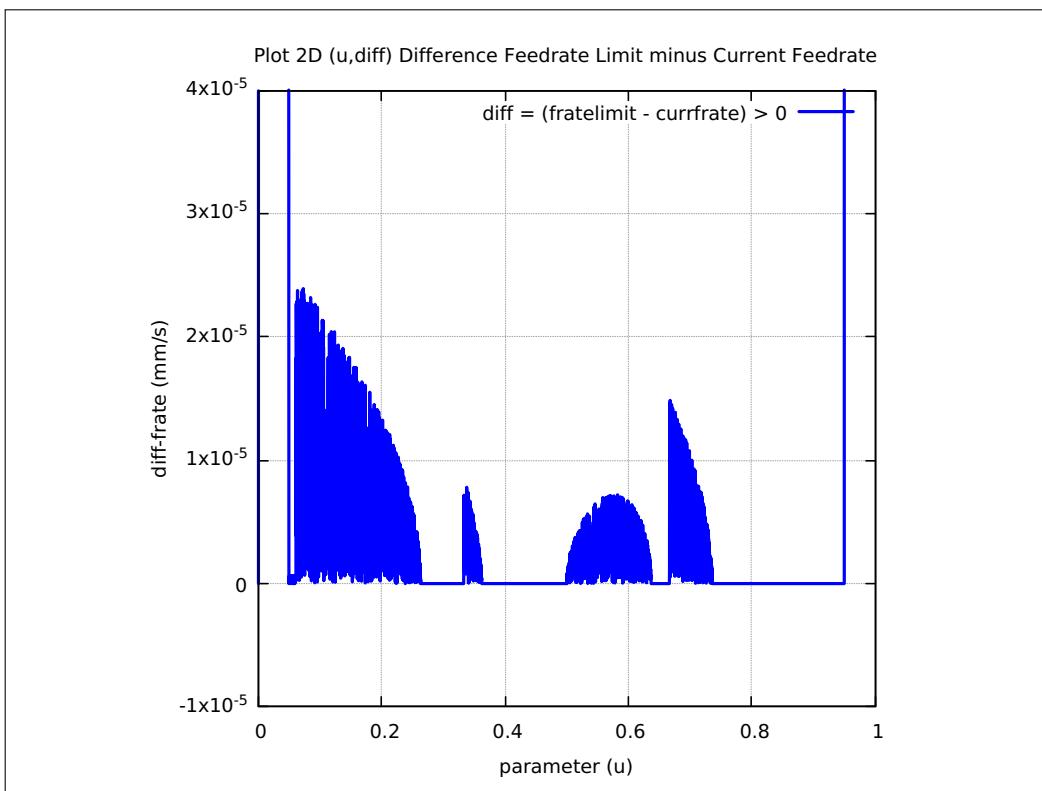


Figure 4.36: Teardrop Current Feedrate Absolute Constraint



4.5.11 Teardrop Histogram distribution of interpolated points

Figure [4.37] shows a histogram for the distribution of interpolated point for the Teardrop curve covering the full range of parameter u , ($u = 0.00 \leq u \leq 1.00$), divided into 10 bins of width 0.10 each.

The histogram was generated for the common lamda = 0.18 and four(4) Feedrate Commands comprising FC10, FC20, FC30 and FC40. The corresponding data table that was used to generate this histogram is provided in the next section.

4.5.12 Teardrop Table distribution of interpolated points

Table [4.3] shows the resulting data for the different Feedrate Command FC, execution runs of the algorithm against the Teardrop curve.

Generally, the total number of interpolated points decreases as the Feedrate Command increases. This can be seen in the last row of the data table. This makes sense since for the same total length of the Teardrop curve, an increase in FC means an increase in feedrate, thus increasing the chord-length of travel during the same interpolation period (constant delta time 0.001 s). This increase in chord-length means a decrease in total interpolated points. The side effect is an increase in the total sum-chord-error since each chord-length increases.

The concept of Total Interpolated Points (TIP) for a single curve like the Teardrop curve does not give much useful information in itself. But when comparisons are made for Total Interpolated Points (TIP) of all ten(10) parametric curves, the interpretation will provide useful information. This is one of the objectives of this thesis. The subject is handled in a later section under Overall Execution Results at Reference [4.8].

Figure 4.37: Teardrop Histogram distribution of interpolated points

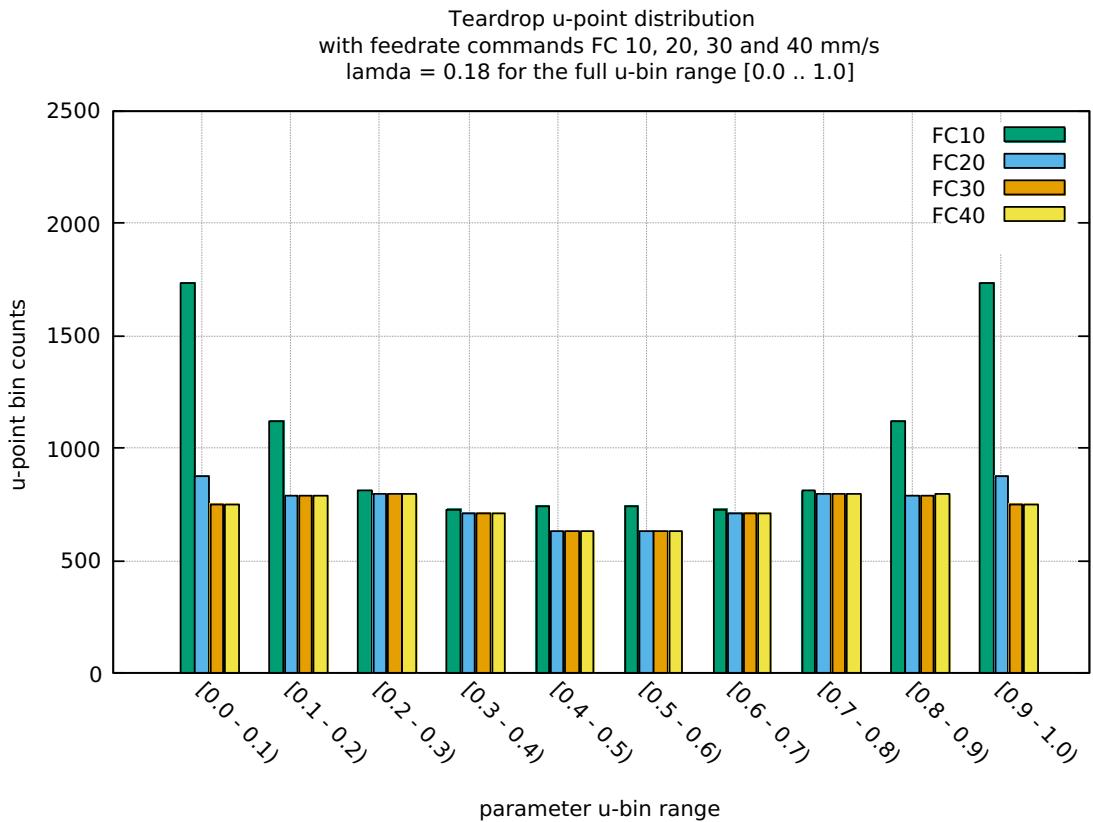


Table 4.3: Teardrop Table distribution of interpolated points

BINS	FC10	FC20	FC30	FC40
0.0 – 0.1	1734	875	748	748
0.1 – 0.2	1120	791	791	791
0.2 – 0.3	809	794	794	794
0.3 – 0.4	726	710	711	711
0.4 – 0.5	741	629	629	629
0.5 – 0.6	742	629	628	629
0.6 – 0.7	726	710	711	711
0.7 – 0.8	809	794	794	793
0.8 – 0.9	1120	791	791	792
0.9 – 1.0	1734	876	750	749
Total counts	10261	7599	7347	7347

4.5.13 Teardrop Rising Current Feedrate Profile

The profiles for the rising feedrate curves are shown in Figure [4.38] on the next page. The parameter range for the S-curve rise is set at $(0.00 \leq u \leq 0.05)$ or 5 percent of the total interpolated points. The lamda value chosen is 0.18 and the Feedrate Command is 20.0 mm/s.

It should be noted that throughout both the rising and falling regions, the S-curve equation is only used to set the Feedrate_Limit(u) and not the Running_Feedrate(u). The algorithm still calculates the running feedrate and the chord-error, during which, both of their constraints are maintained. This ensures that there are no exceptions in the algorithm that both or either one of the constraints are violated. Both must not be violated, absolutely.

4.5.14 Teardrop Falling Current Feedrate Profile

The profiles for the falling feedrate curves are shown in Figure [4.39] on the next page. The parameter range for the S-curve fall is set at $(0.95 \leq u \leq 1.00)$ or also 5 percent of the total interpolated points. The lamda value chosen is 0.18 and the Feedrate Command is 20.0 mm/s.

Figure 4.38: Teardrop Rising Current Feedrate Profile

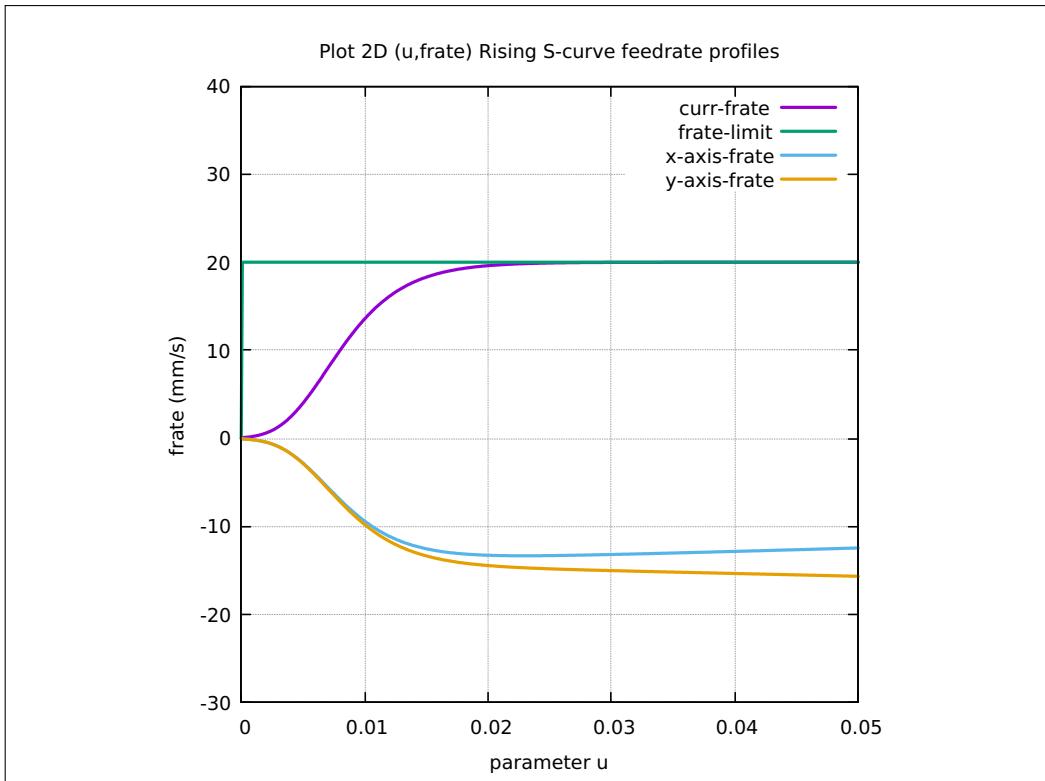
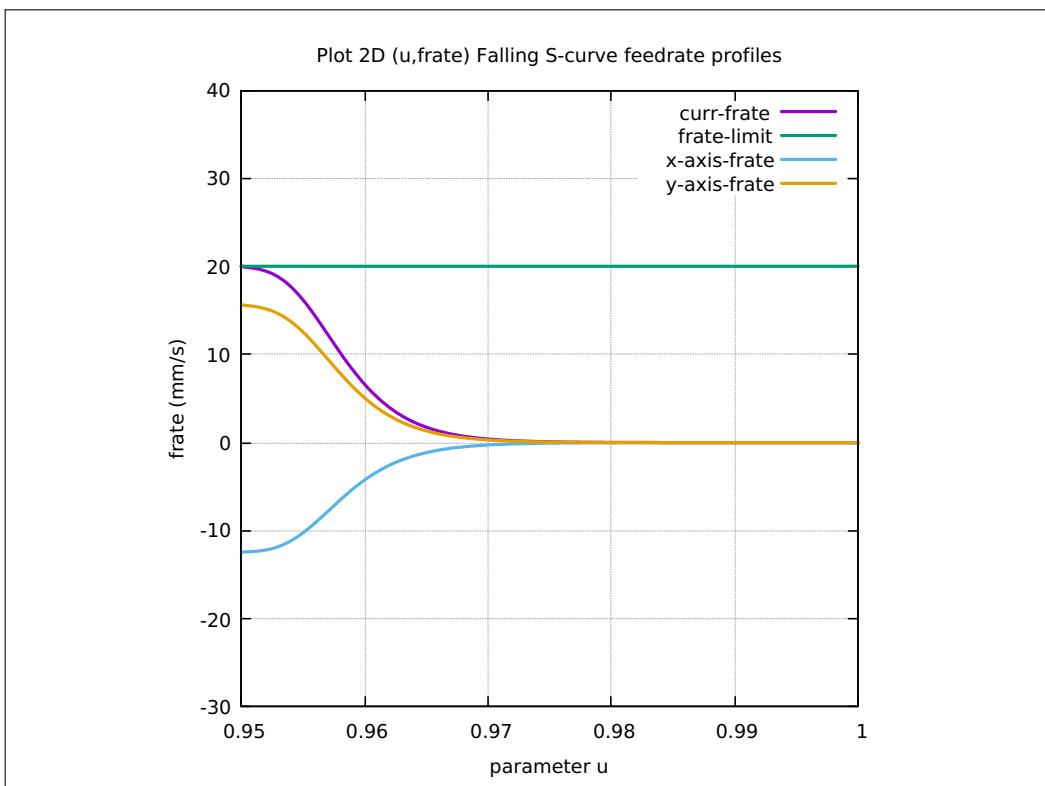


Figure 4.39: Teardrop Falling Current Feedrate Profile



4.5.15 Teardrop FC10, FC20, TC30 & FC40 Running Feedrates

The next two(2) pages show four(4) execution results for the Teardrop curve at $\lambda = 0.18$ for feedrate commands FC10, FC20, FC30 and FC40, in Figure [4.40], Figure [4.41], Figure [4.42] and Figure [4.43], respectively.

The next four(4) plots show that all of the running feedrates remain below the respective feedrate command FC. This is a constraint that cannot be violated in all circumstances.

These plots cover the feedrate rising region (5 % points), the main feedrate activity region (90 % points), and the feedrate falling region (5 % points).

The first plot for FC10, shows that a significant portion of the current (running) feedrate flattens at the 10 mm/s level (FC10). The second plot for FC20 shows a small portion of the current feedrate flattens at the 20 mm/s level (FC20). In the case of both the third and fourth plots (FC30 and FC40) respectively, their current feedrates never reach their set Feedrate Command levels.

It is important to remember that the feedrate limit is different from the current or running feedrate. Both feedrates are nearly equal in value, with running feedrate always just a little lower than the feedrate limit. Both feedrates vary with parameter u , unlike the feedrate command (FC), which is a constant. They are not displayed in the plots because they essentially overlap due to a relatively large y-scale display.

The x-axis and y-axis feedrates are self-explanatory. A negative value means moving in the opposite direction. The magnitude of the feedrate (a scalar) remains the same.

Figure 4.40: Teardrop FC10 Running Feedrate Profiles

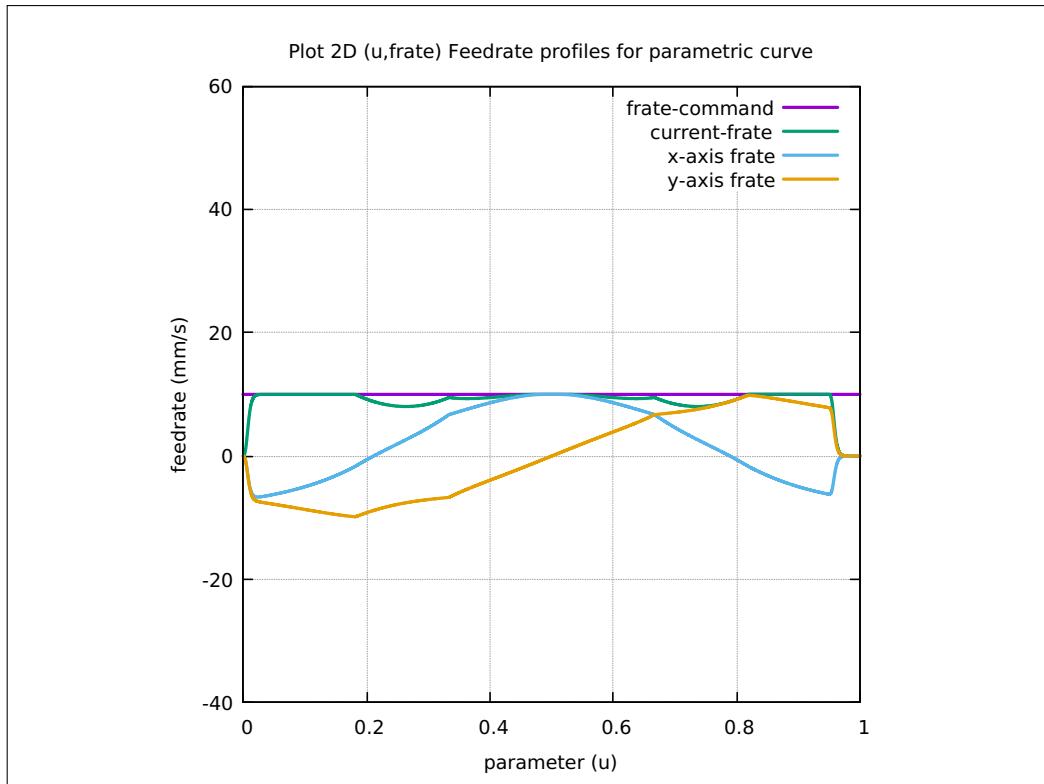


Figure 4.41: Teardrop FC20 Running Feedrate Profiles

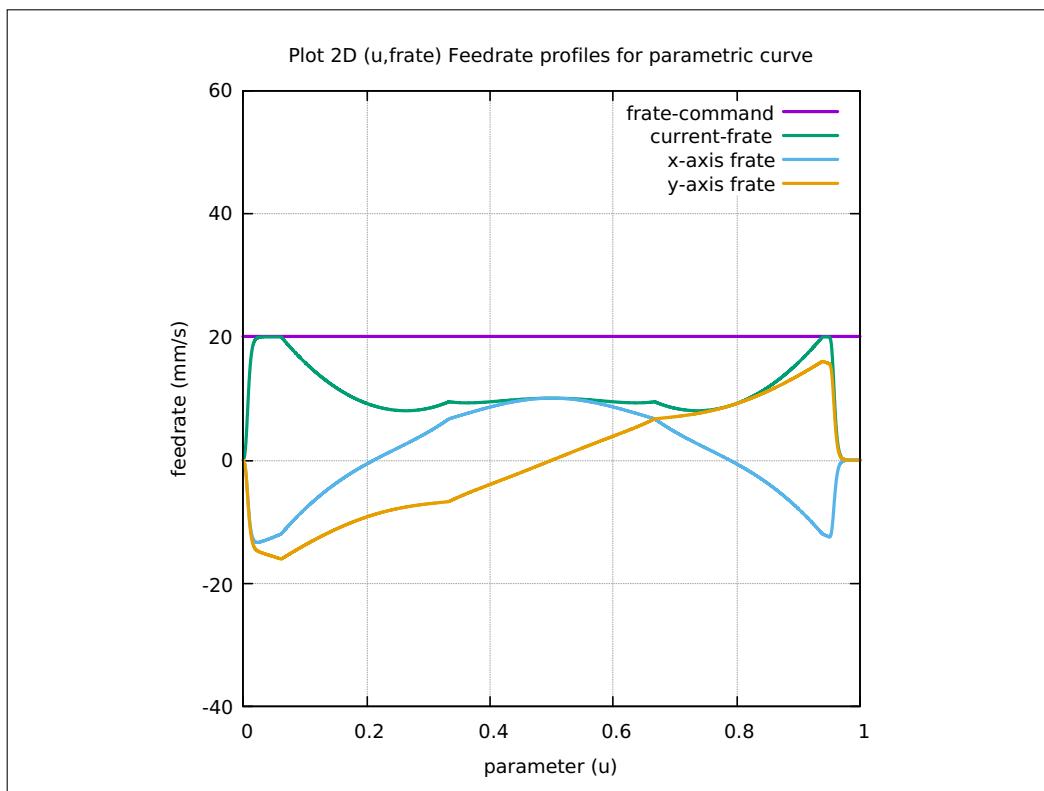


Figure 4.42: Teardrop FC30 Running Feedrate Profiles

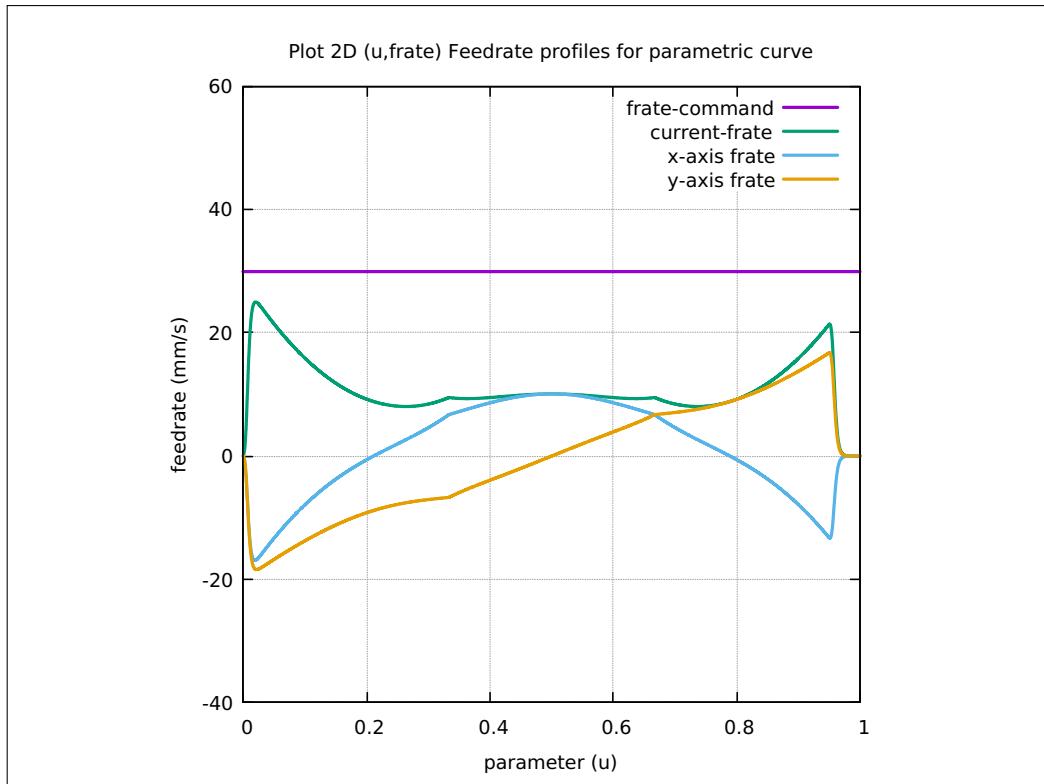
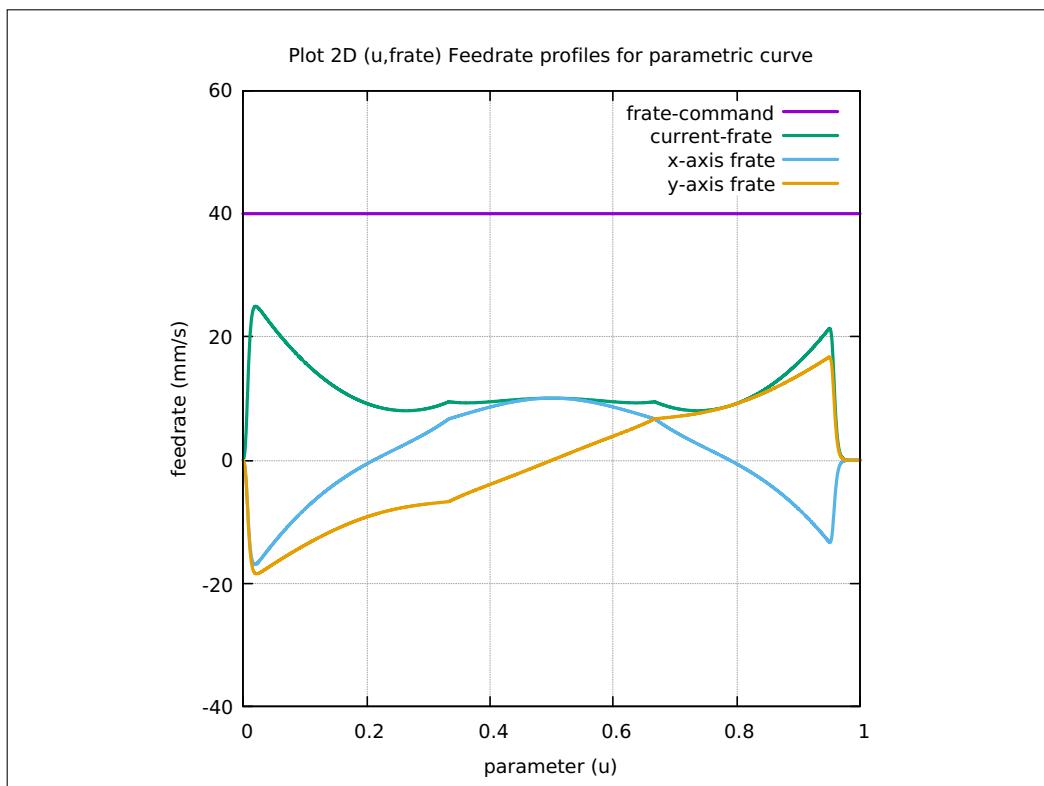


Figure 4.43: Teardrop FC40 Running Feedrate Profiles



4.5.16 Teardrop Color-coded Running Feedrates

In the next two(2) portrait pages, the Teardrop curve colored feedrate execution profiles for FC10 in Figure [4.44], FC20 in Figure [4.45], FC30 in Figure [4.46] and FC40 in Figure [4.47], respectively, are displayed.

The color spectrum legend on the right column for each curve was adjusted so that the maximum color code (red) represents the Feedrate Command FC for the particular run execution. This is the maximum value of the achievable running feedrate for each run. This setting provides more variations in colors along the curve path for each individual plot. Otherwise, the variations are not easily visible if the plots use a common maximum value for the color spectrum code.

As can be seen in all four(4) plots, the smoothness in running feedrates throughout the entire full curve path is shown by the linear and gradual transitions of colors (increasing or decreasing feedrates), that follow exactly the transitions in the color spectrum scale. Essentially, there are no sudden jump in colors.

This smoothness requirement on running feedrate is one of the main objectives of the algorithm in this thesis.

Figure 4.44: Teardrop FC10 Colored Feedrate Run Profile

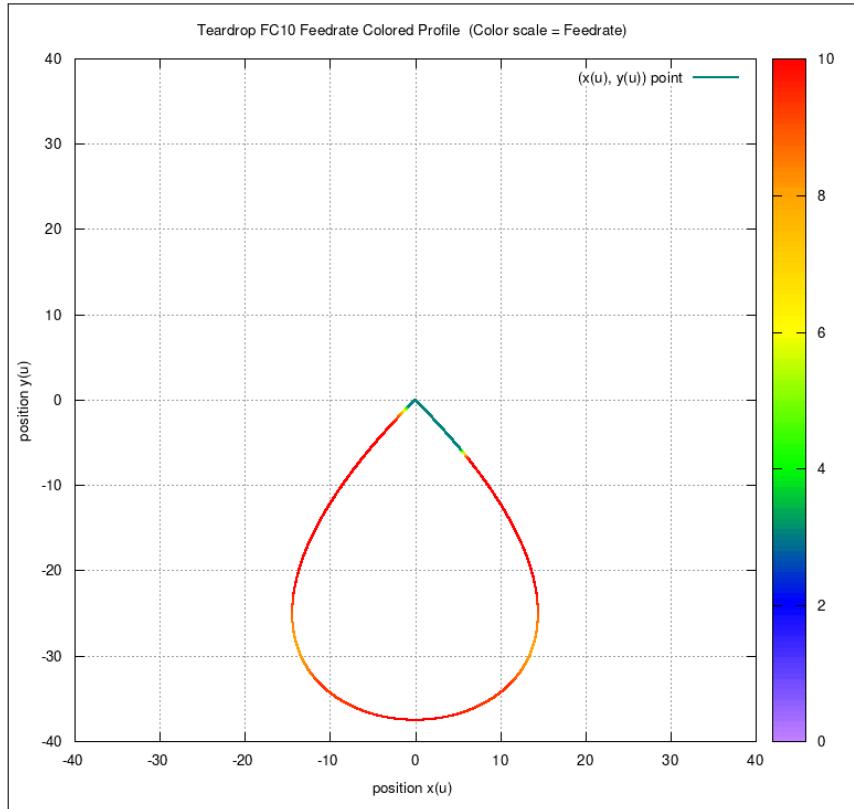


Figure 4.45: Teardrop FC20 Colored Feedrate Run Profile

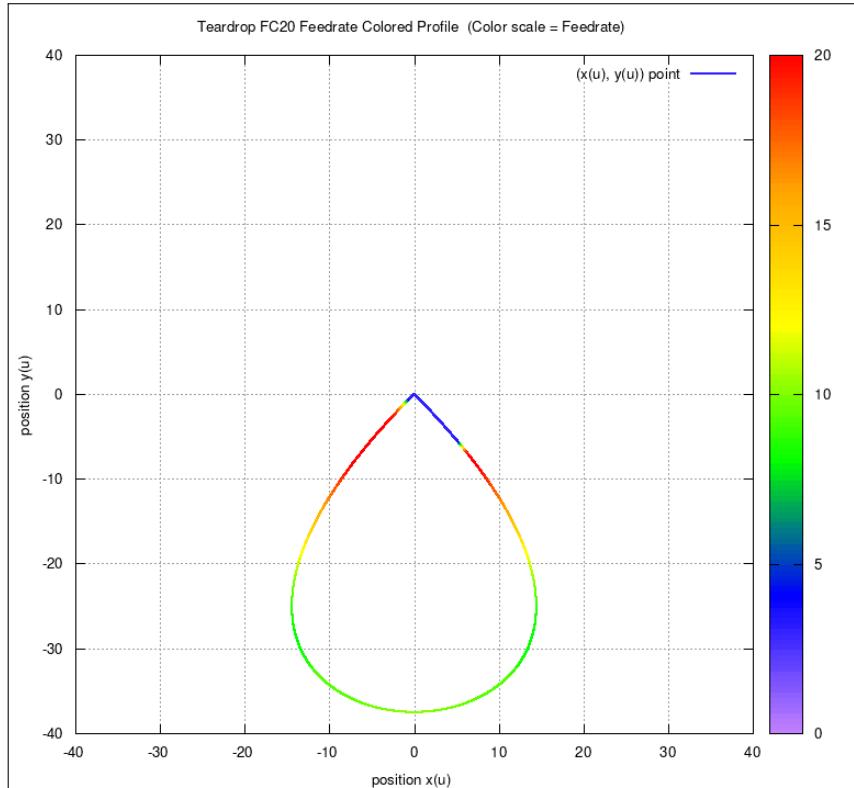


Figure 4.46: Teardrop FC30 Colored Feedrate Run Profile

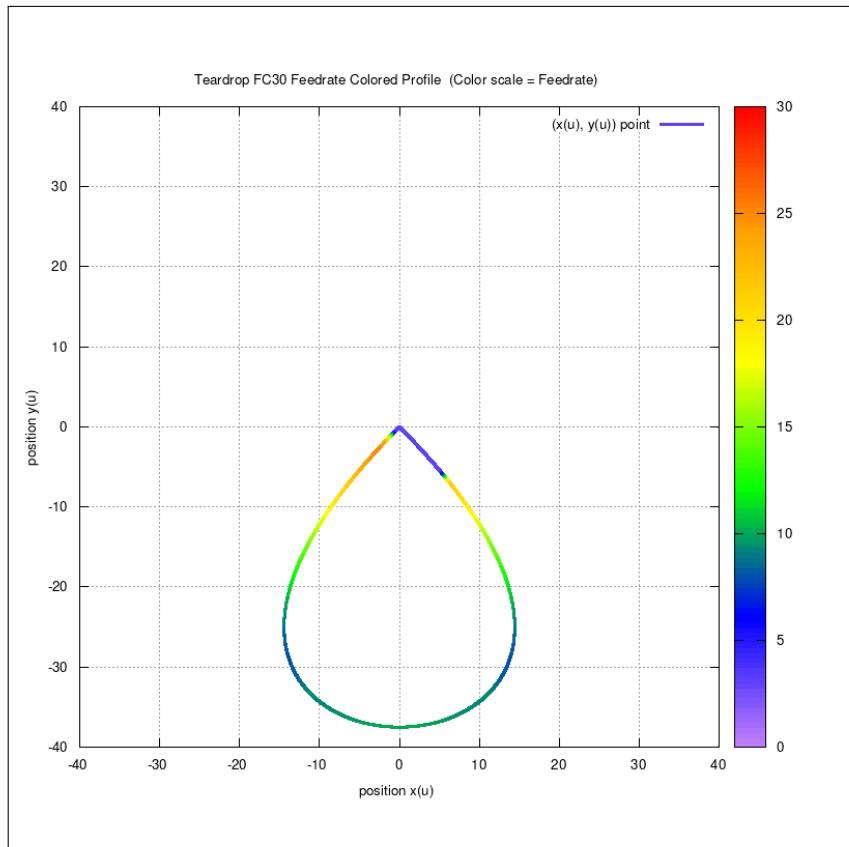
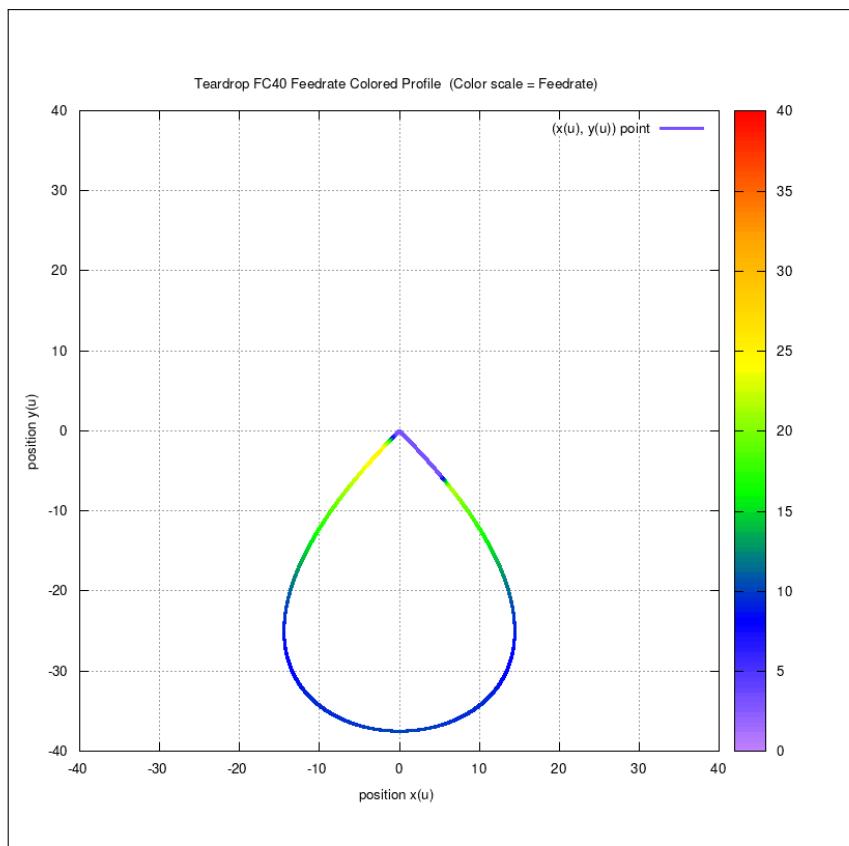


Figure 4.47: Teardrop FC40 Colored Feedrate Run Profile



4.5.17 Teardrop-Table FC 10, 20, 30 & 40 Performance data

For the overall interpolation algorithm performance of the Teardrop curve, 20 different metrics were considered. The results are provided in Table [4.5], displayed in landscape mode.

The first three(3) rows of mandatory input parameters (Curve Type, Feedrate Command FC, and Lamda) uniquely identifies an execution run of the interpolation algorithm. The terms used in the data table are described as follows.

Table 4.4: Terms used in Curve Performance data

TIP	Total Interpolated Points	Total number of points generated on the resulting curve by the interpolation algorithm for a complete run from $u = 0.00$ until $u = 1.00$.
SAL	Sum Arc Length	Sum of the arc-lengths accumulated as the parameter u is incremented to $u + (u\text{-next})$.
SAT	Sum Arc Theta	Sum of the arc-theta angles in radians accumulated as the parameter u is incremented to $u + (u\text{-next})$. The arc-theta is the angle subtended by the arc segment between $\rho(u)$ and $\rho(u + (u\text{-next}))$, where $\rho(u)$ is the Radius of Curvature at the point u .
SAA	Sum Arc Area	Sum of the arc-areas accumulated as the parameter u is incremented to $u + (u\text{-next})$. The arc-area is the area bordered by the chord from point u to $u + (u\text{-next})$, and the corresponding arc segment.
SCL	Sum Chord Length	Sum of the chord-lengths accumulated as the parameter u is incremented to $u + (u\text{-next})$.
SCE	Sum Chord Error	Sum of the chord-errors accumulated as the parameter u is incremented to $u + (u\text{-next})$.
AAL	Average Arc Length	$AAL = (SAL)/(TIP-1)$
AAT	Average Arc Theta	$AAT = (SAT)/(TIP-1)$
AAA	Average Arc Area	$AAA = (SAA)/(TIP-1)$
ACL	Average Chord Length	$ACL = (SCL)/(TIP-1)$
ACE	Average Chord Error	$ACE = (SCE)/(TIP-1)$
RA1	(SCE/TIP)	Sum Chord Error divided by Total Interpolated Points
RA2	(SCE/SCL)	Sum Chord Error divided by Sum Chord Length
RA2	(SAA/SCL)	Sum Arc Area divided by Sum Chord Length

Row (4) - Generally, as the Feedrate Command FC increases the number of Total Interpolated Points (TIP) decreases as expected. The last two columns FC30 and FC40 do not show reduction because the algorithm could not optimize constraints of chord-error and feedrate any further.

Row (5) - When the Feedrate Command FC increases, the Sum-Chord-Error (SCE) also increases. This is expected, since as FC increases, the chord-length increases, and with bigger (longer) chord-length the bigger the chord-error.

Row (6) - (SCE/TIP-1) which is Sum-Chord-Error divided by the number of chords for the entire length of the curve provides indicative comparison measures against the number of interpolated points for FC10, FC20, FC30 and FC40.

Rows (7) and (8) - These two rows, SAL and SCL, are interesting because SAL refers to the sum of arc-lengths while SCL refers to the sum of chord-lengths. The difference cannot be zero, no matter how small it is because of its geometrical definitions. SAL (arc-length) must always be greater than SCL (chord length) since individual chord lengths cannot be greater than its arc-length. If they are equal, then both are straight lines, and there is no arc segment anymore.

It is important to note that the calculation of the sum arc-length (SAL) in this work is just approximate. It is good for a circle because of the circle's perfect shape. For general curves, it is just a first-order approximation. In fact, a large body of work is involved in what is termed as "Near-Arc Approximation" for the calculation of the sum arc length (SAL). This method is not covered in this work.

The special cases of the Circle and Ellipse curves, regarding SAL and SCL will be discussed separately in this Chapter 4, under section Notable Results Rest of Curves with reference link [4.7].

Between SAL and SCL, the more reliable calculation is SCL, because the formula used for chord-length calculation (SCL) is exact. It is just the length of the line connecting two (x,y) points on the parametric curve.

Row (9) - The difference (SAL-SCL) must always be positive for all Feedrate Commands FC. However, for the Teardrop curve there is a negative result in column 3, for FC30. Similarly, this issue will be discussed further in this chapter, under section Notable Results Rest of Curves with reference link [4.7].

Row (10) - The percentage difference (SAL-SCL)/SAL is important because it provides a value that is relative to its own size (length of curve). The difference (SAL - SCL) value alone is not meaningful because it varies differently for different curves. So percentage information is important because the comparison can be used against curves of different sizes (lengths).

Row (11) - This metric (SCE/SCL), which is the total sum of chord-errors divided by the total sum of chord-lengths, is the most meaningful measure of the efficiency or effectiveness of the algorithm developed in this work. This metric is general and not specific to any parametric curve. It is considered the performance measure for algorithm efficiency because it is independent of curve length.

The situation is analogous to comparing running speeds in meters per second. Whether you can run fast or not depends on your speed, and it does not matter how far or long you run. In the case of running it is about how many meters can you cover in one second. In the case of this work, it is about how much error does the algorithm generate per unit length of the curve traversed.

Row (12) and Row (13) - The meanings of Sum-Arc-Theta (SAT) and Sum-Arc-

Area (SAA) are straightforward and self-explanatory.

Row (14) - This metric (SAA/SCL) or Sum Arc Area divided by the Sum Arc Length, carries the meaning similar to (SCE/SCL) in Row (11). However, the calculation of Arc-Area is approximate and not reliable for general curves, except for the Circle and Ellipse, which are perfect shapes. This is due to the fact that the Radius of Curvature $\rho(u)$, is used in the arc-area calculation. Similarly, this issue will be discussed further in this chapter, under section Notable Results Rest of Curves with reference link [4.7].

Rows (15) to (19) - These rows are averages and they are self-explanatory. These averages provide valuable information when comparing among different parametric curves. This issue will be discussed further in this chapter, under section Overall Execution Results with reference link [4.8].

Row (20) - This row Actual Runtime (ART), execution of the algorithm on the computer, measured in seconds, is just an indication (not exact measurements) regarding how many internal computing operations were carried out in FC10, FC20, FC30 and FC40 runs. Generally, algorithm runtime increases as the Feedrate Command FC increases. It should be remembered that the algorithm performs recursive and iterative computations in order to constrain both chord-error and feedrate simultaneously. It means that the higher the feedrate command, the more internal computations the algorithm has to perform.

Table 4.5: Teardrop Table FC10-20-30-40 Run Performance data

1	Curve Type	TEARDROP	TEARDROP	TEARDROP
2	User Feedrate Command FC(mm/s)	FC10	FC20	FC40
3	User Lamda Acceleration Safety Factor	0.18	0.18	0.18
4	Total Interpolated Points (TIP)	10261	7599	7347
5	Total Sum-Chord-Error (SCE) (mm)	5.809737838076E-03	7.140807162860E-03	7.336793707381E-03
6	Ratio 1 = (SCE/TIP) = Chord-Error/Point	5.662512512745E-07	9.39827212807E-07	9.987467611463E-07
7	Total Sum-Arc-Length (SAL) (mm)	1.018356741269E+02	1.018418663504E+02	1.018595636256E+02
8	Total Sum-Chord-Length (SCL) (mm)	1.018356732173E+02	1.018418655699E+02	1.018595666011E+02
9	Difference = (SAL - SCL) (mm)	9.095903408252E-07	7.805327442156E-07	-2.975420997586E-06
10	Percentage Difference = (SAL - SCL)/SAL	8.931942058845E-07	7.664163788301E-07	-2.921101261067E-06
11	Ratio 2 = (SCE/SCL) = Chord Error/Chord-Length	5.705012452441E-05	7.011661778683E-05	7.202851879506E-05
12	Total Sum-Arc-Theta (SAT) (rad)	4.712304324578E+00	4.712268805770E+00	4.712349787227E+00
13	Total Sum-Arc-Area (SAA) (mm2)	3.822127588087E-05	6.182290957317E-05	6.781012634326E-05
14	Ratio 3 = (SAA/SCL) = Arc-Area/Chord-Length	5.705012452441E-05	7.011661778683E-05	7.202851879506E-05
15	Average-Chord-Error (ACE) (mm)	5.662512512745E-07	9.39827212807E-07	9.987467611463E-07
16	Average-Arc-Length (AAL) (mm)	9.925504300871E-03	1.340377288107E-02	1.386599014779E-02
17	Average-Chord-Length (ACL) (mm)	9.925504212216E-03	1.340377277834E-02	1.386599055283E-02
18	Average-Arc-Theta (AAT) (rad)	4.592889205241E-04	6.201985793327E-04	6.414851330284E-04
19	Average-Arc-Area (AAA) (mm2)	3.725270553691E-09	8.136734610840E-09	9.230891143923E-09
20	Algorithm actual runtime on computer (ART) (s)	4.487434922	19.907344569	28.094412173
				32.96324077

4.5.18 Teardrop Algorithm Performance and its metrics

The next four(4) portrait pages involve algorithm performance characteristics on the Teardrop curve. There are four(4) metrics developed to assess algorithm performance. The performance metrics for the Teardrop curve are described as follows.

(1) Performance metric 1 (SCE/TIP) : This is the ratio of total sum-chord-error divided by the total number of interpolated points. The histogram plot is provided in Figure [4.48] and its corresponding data is provided in Table [4.6]. This metric provides an impression on the average generated chord-error per chord. The number of chords is the Total Interpolated Points minus 1.

(2) Performance metric 2 (SCE/SCL) : This is the ratio of total sum-chord-error divided by the total sum-chord-length. The histogram plot is provided in Figure [4.49] and its corresponding data is provided in Table [4.7]. This metric accurately reflects the amount of chord-error generated per unit length of chord traversed. This is the most useful and meaningful metric for the assessment of algorithm performance.

(3) Performance metric 3 (SAA/SCL) : This is the ratio of the sum-arc-areas divided by the total sum-chord-length. The histogram plot is provided in Figure [4.50] and its corresponding data is provided in Table [4.8]. This metric is useful and meaningful when the arc-area calculations are accurate, for example, in cases of the circle and ellipse curves. For the rest, this metric is suspect due to the crude approximation in the arc-area calculation.

(4) Performance metric 4 ((SAL-SCL)/SAL)*100 : This is the ratio of the difference between the sum-arc-length and the sum-chord-length, divided by the sum-arc-length and multiplied by 100 to represent it in percentage form. The histogram plot is provided in Figure [4.51] and its corresponding data is provided in Table [4.9]. For some of the curves where the calculation of arc-area is not reliable, this metric is not reliable.

Figure 4.48: Teardrop Run Performance 1: SCE/TIP

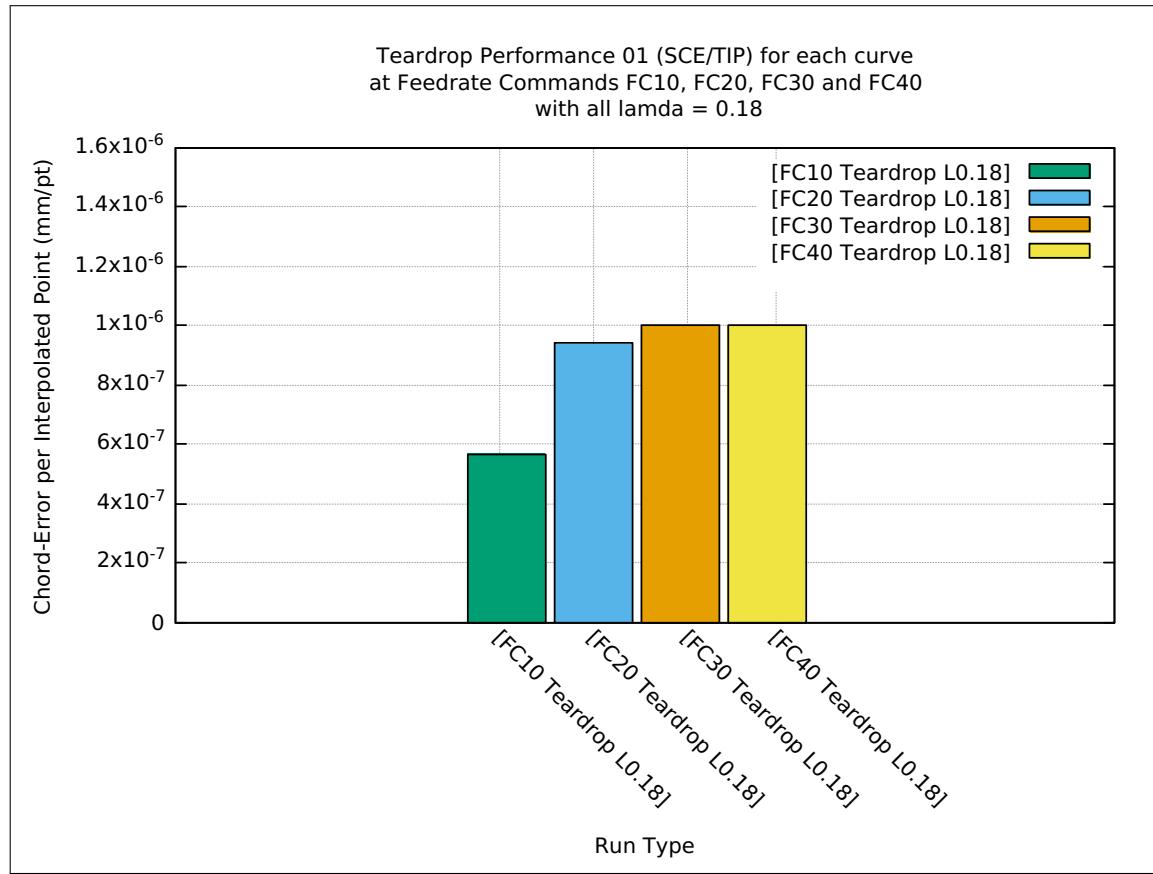


Table 4.6: Teardrop SCE/TIP Table FC10-20-30-40 Run Performance data

Teardrop Performance 1: (SCE/TIP)

Chord-Error per Interpolated-Point

FC10 Teardrop L0.18	5.662512512745E-07
FC20 Teardrop L0.18	9.398272128007E-07
FC30 Teardrop L0.18	9.987467611463E-07
FC40 Teardrop L0.18	9.985225973861E-07

Figure 4.49: Teardrop Run Performance 2: SCE/SCL

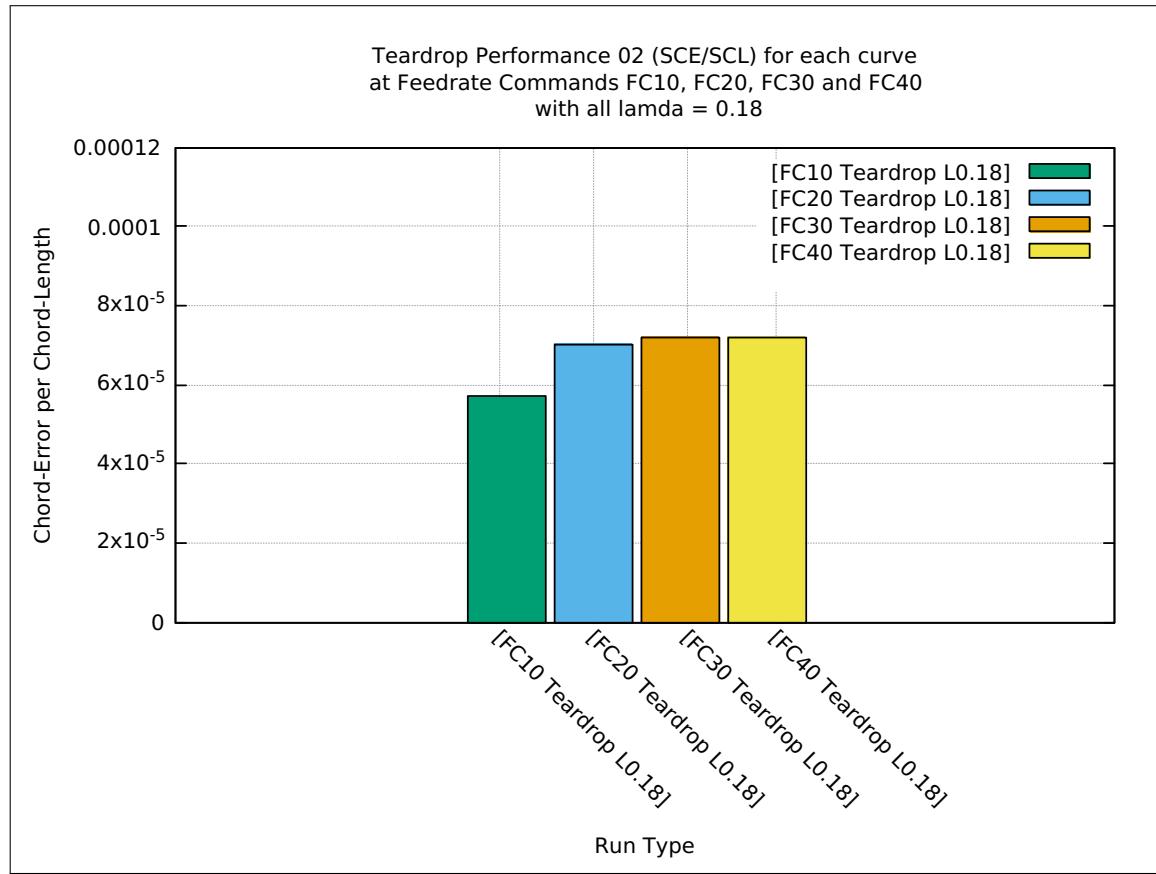


Table 4.7: Teardrop SCE/SCL Table FC10-20-30-40 Run Performance data

Teardrop Performance 2:
(SCE/SCL)

Chord-Error per Chord-Length

FC10 Teardrop L0.18	5.705012452441E-05
FC20 Teardrop L0.18	7.011661778683E-05
FC30 Teardrop L0.18	7.202851879506E-05
FC40 Teardrop L0.18	7.202932786929E-05

Figure 4.50: Teardrop Run Performance 3: SAA/SCL

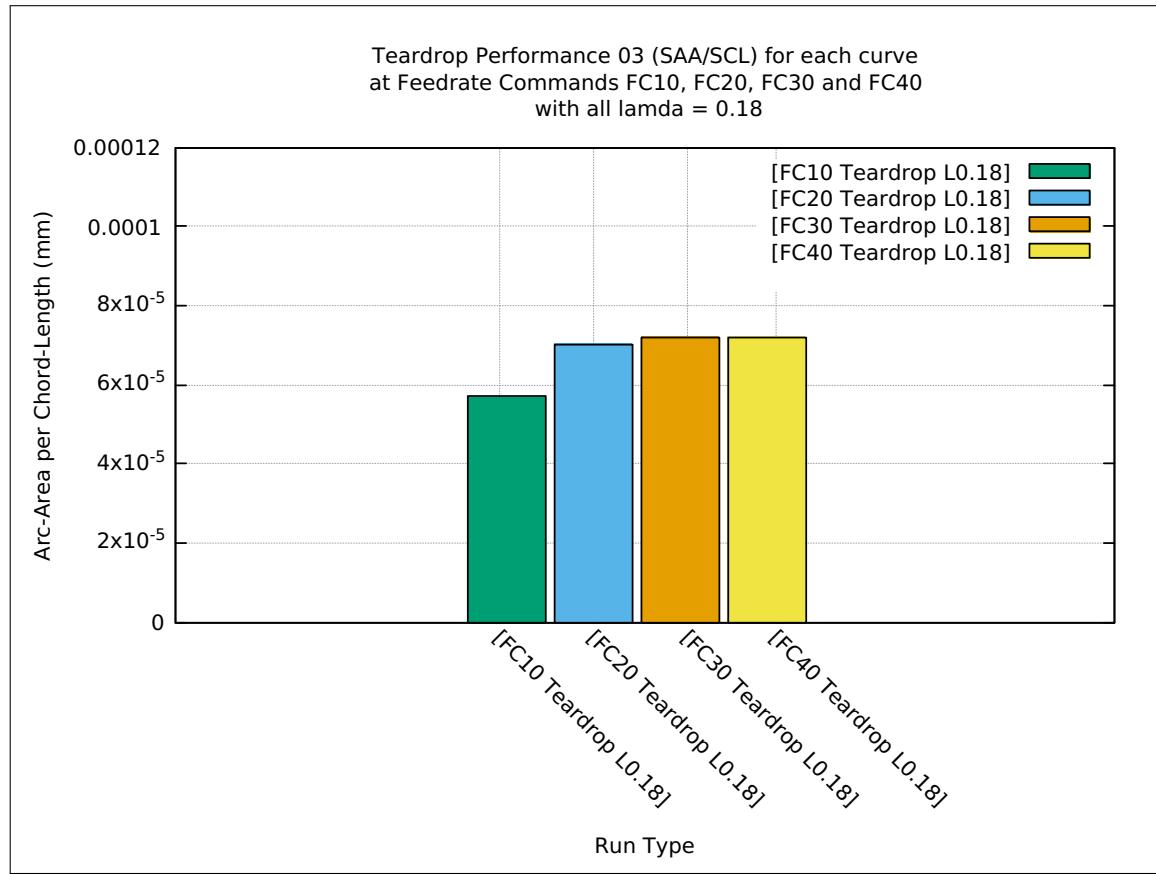


Table 4.8: Teardrop SAA/SCL Table FC10-20-30-40 Run Performance data

Teardrop Performance 3: (SAA/SCL)

Arc-Area per Chord-Length

FC10 Teardrop L0.18	5.705012452441E-05
FC20 Teardrop L0.18	7.011661778683E-05
FC30 Teardrop L0.18	7.202851879506E-05
FC40 Teardrop L0.18	7.202932786929E-05

Figure 4.51: Teardrop Run Performance 4: $100*(\text{SAL}-\text{SCL})/\text{SAL}$

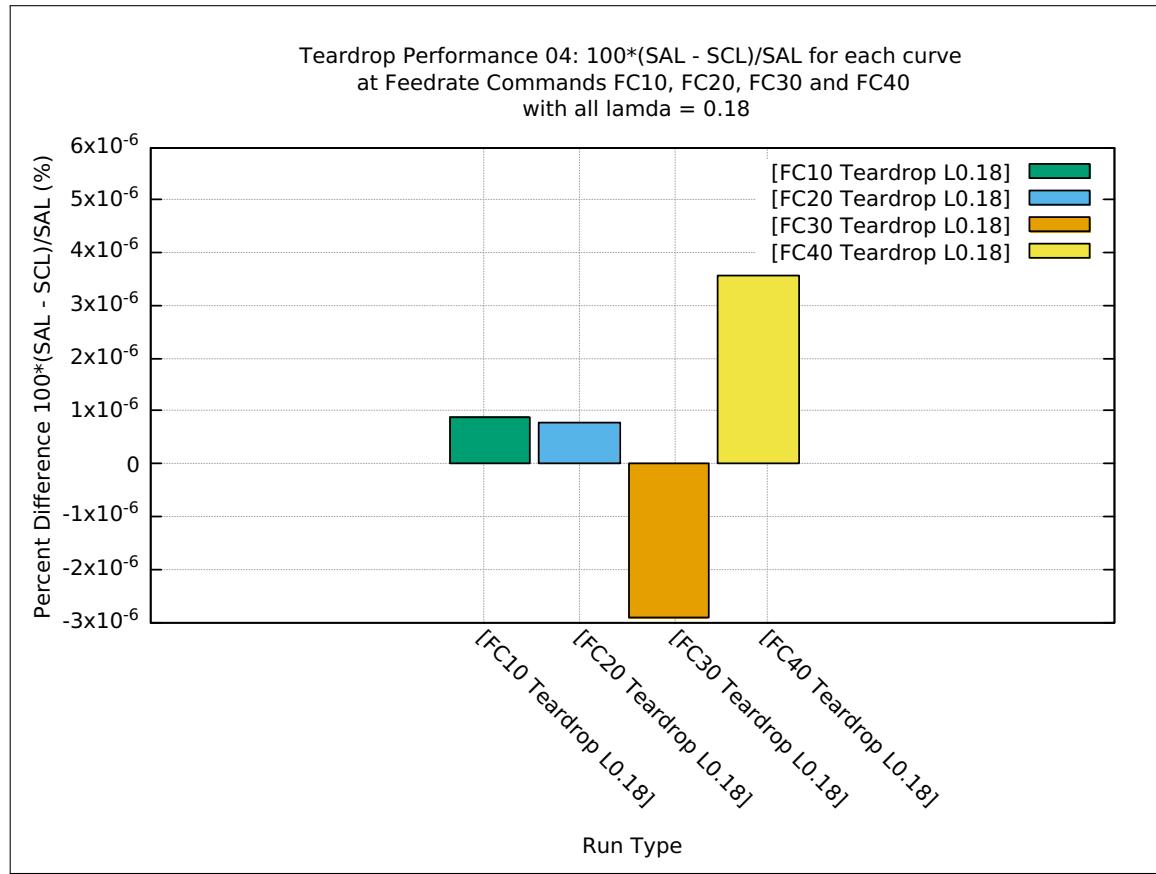


Table 4.9: Teardrop (SAL-SCL)/SAL Table FC10-20-30-40 Run Performance data

Teardrop Percentage Difference : Arc-Length vs Chord-Length	(SAL-SCL)/(SAL)
FC10 Teardrop L0.18	$8.931942058845 \times 10^{-7}$
FC20 Teardrop L0.18	$7.664163788301 \times 10^{-7}$
FC30 Teardrop L0.18	$-2.921101261067 \times 10^{-6}$
FC40 Teardrop L0.18	$3.55205036776 \times 10^{-6}$

Negative values for this metric shall be discussed in this chapter, under section Notable Results Rest of Curves with reference link [4.7].

4.5.19 Teardrop Tangential Acceleration Run Profiles

In the next four(4) landscape pages, the Teardrop curve tangential acceleration $Tang_Accn(u)$, execution profiles for FC10 in Figure [4.53], FC20 in Figure [4.54], FC30 in Figure [4.55] and FC40 in Figure [4.56], respectively, are displayed.

The CNC machine run parameters for the maximum acceleration settings is +30 mm/s² for the x-axis direction, and also +30 mm/s² for the y-axis direction. Similarly, the minimum settings are -30 mm/s² for both the x and y axes, respectively. It is important to note here that negative means simply deceleration or decreasing feedrate.

By Pythagoras Theorem, the maximum magnitude of the acceleration is:

$$(Accn_{max} = \sqrt{30^2 + 30^2}) = \sqrt{1800} = 42.426 \text{ mm/s}^2.$$

This is the value constraining the positive and negative accelerations in the ensuing plots.

The algorithm execution results for the Teardrop curve at FC10, FC20, FC30 and FC40 in the next four(4) pages, show that the tangential acceleration upper and lower bounds ($-42.426 \leq Tang_Accn(u) \leq +42.426$) mm/s², are not violated.

The "vertical drops and rises" seen in tangential acceleration profile are not actually sudden drops or rises since the number of interpolated points around these drops and rises are many, for example, an average of 600 points within $\Delta u = 0.10$ throughout the full curve. Upon closer inspection, as shown in Figure [4.52] for ranges ($0.333 \leq u \leq 0.335$) and ($0.660 \leq u \leq 0.670$), it is clear that these are not sudden drops or rises. This fact negates (denies) the occurrence of acceleration jitter.

The absolute confinement of tangential acceleration is one of the main objectives of the interpolation algorithm in this thesis.

Figure 4.52: Teardrop Closeup-View No Jitters FC40 Tangential Acceleration Run Profile

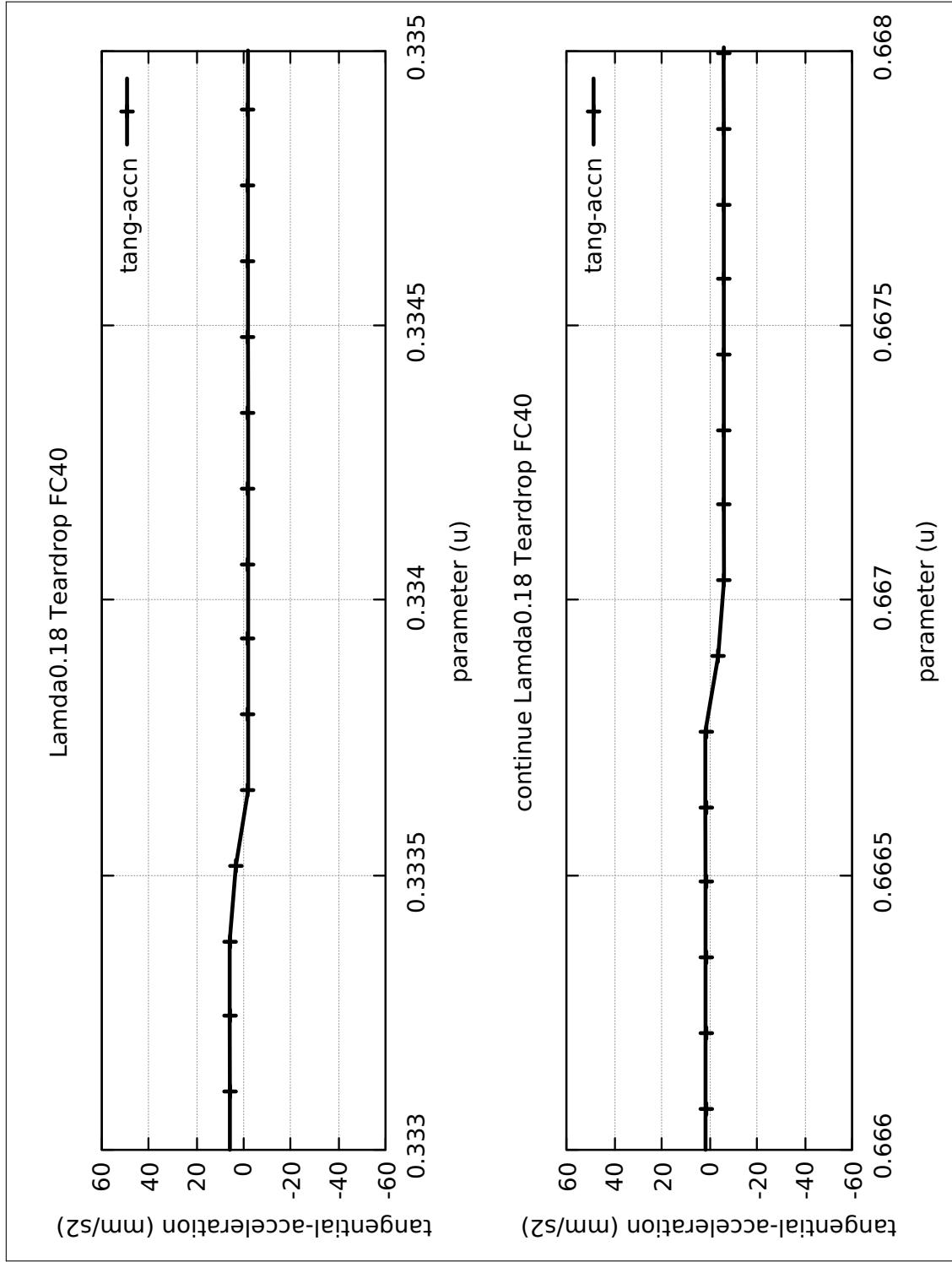


Figure 4.53: Teardrop FC10 Tangential Acceleration Run Profile

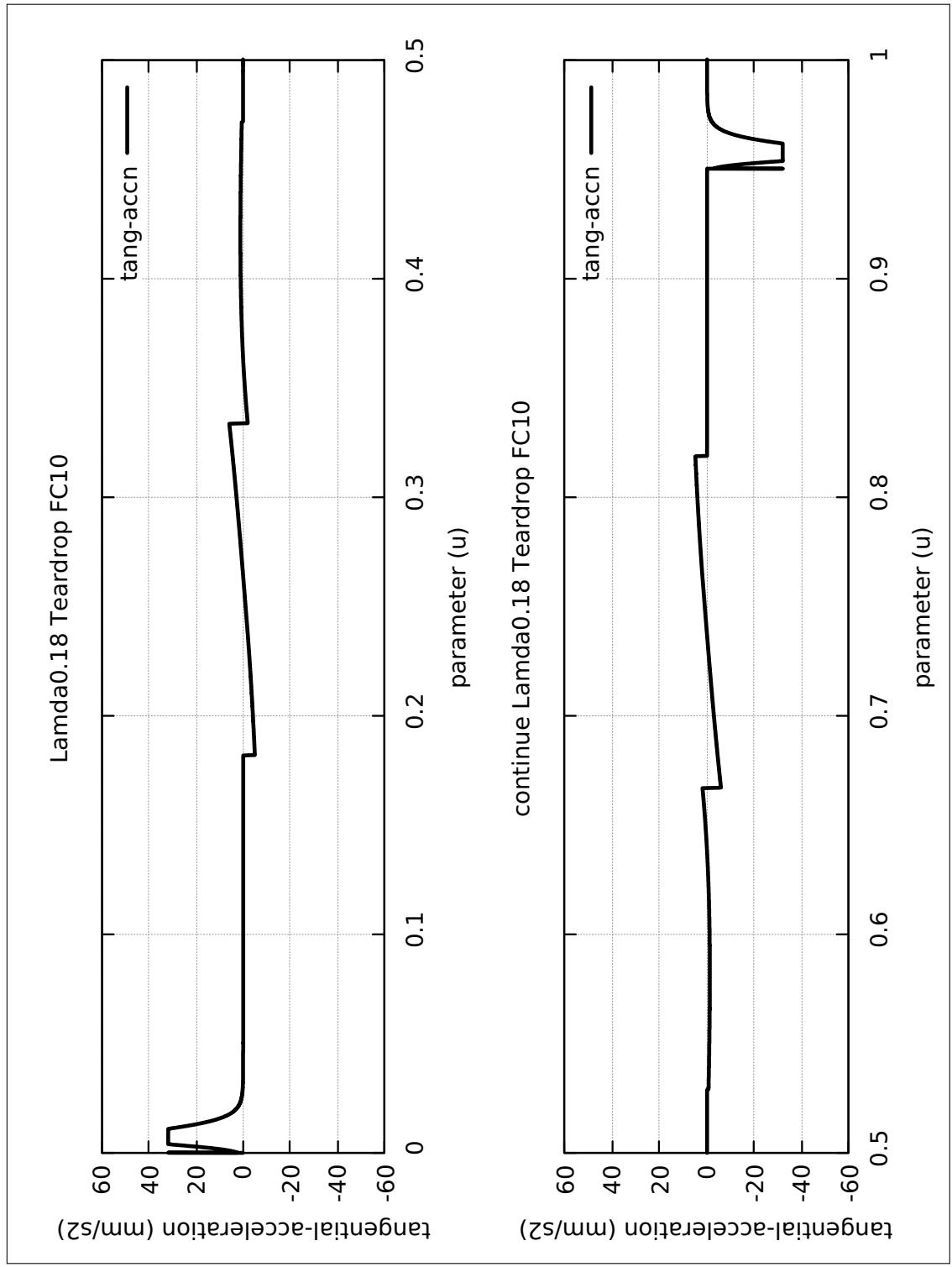


Figure 4.54: Teardrop FC20 Tangential Acceleration Run Profile

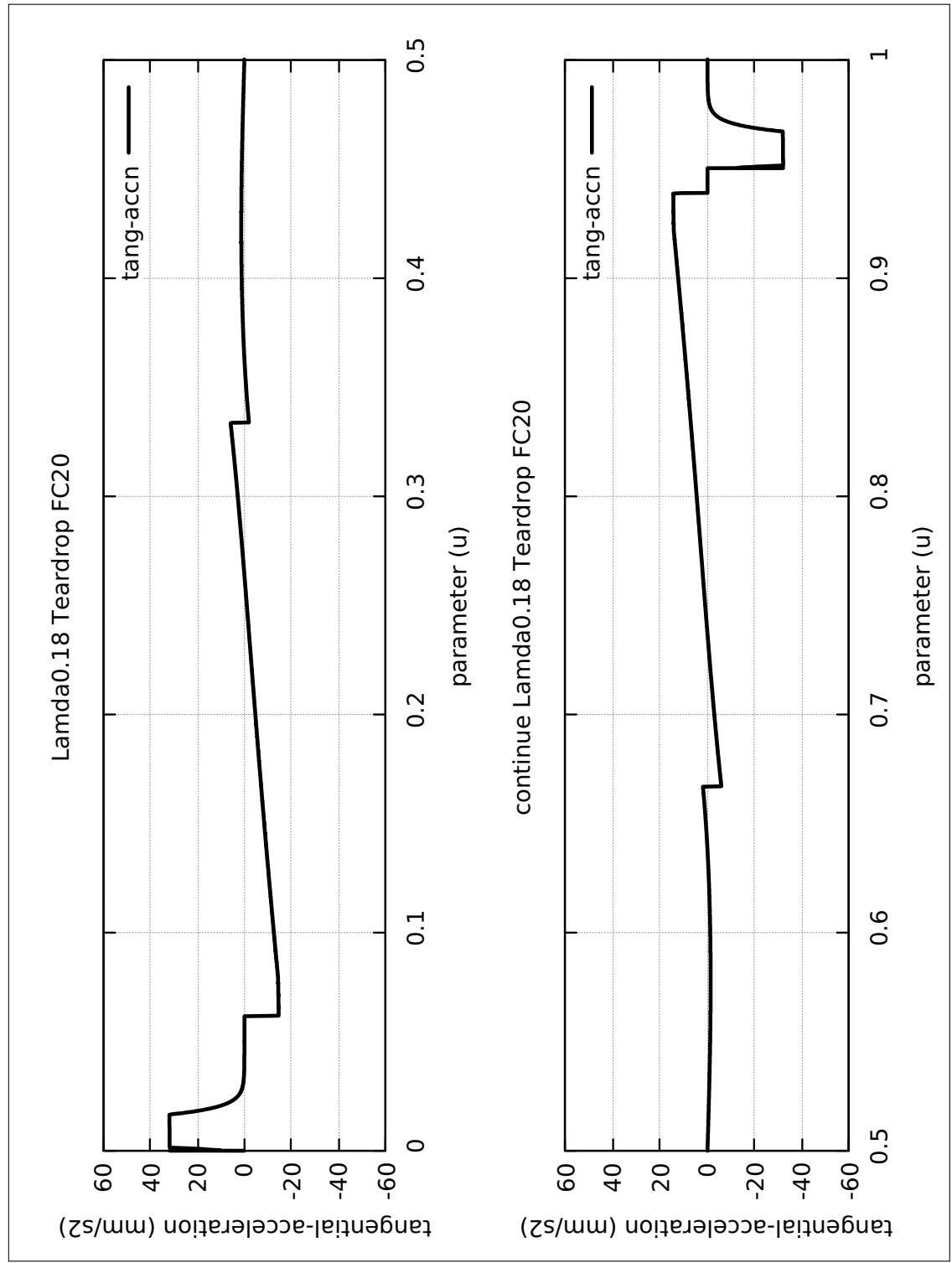


Figure 4.55: Teardrop FC30 Tangential Acceleration Run Profile

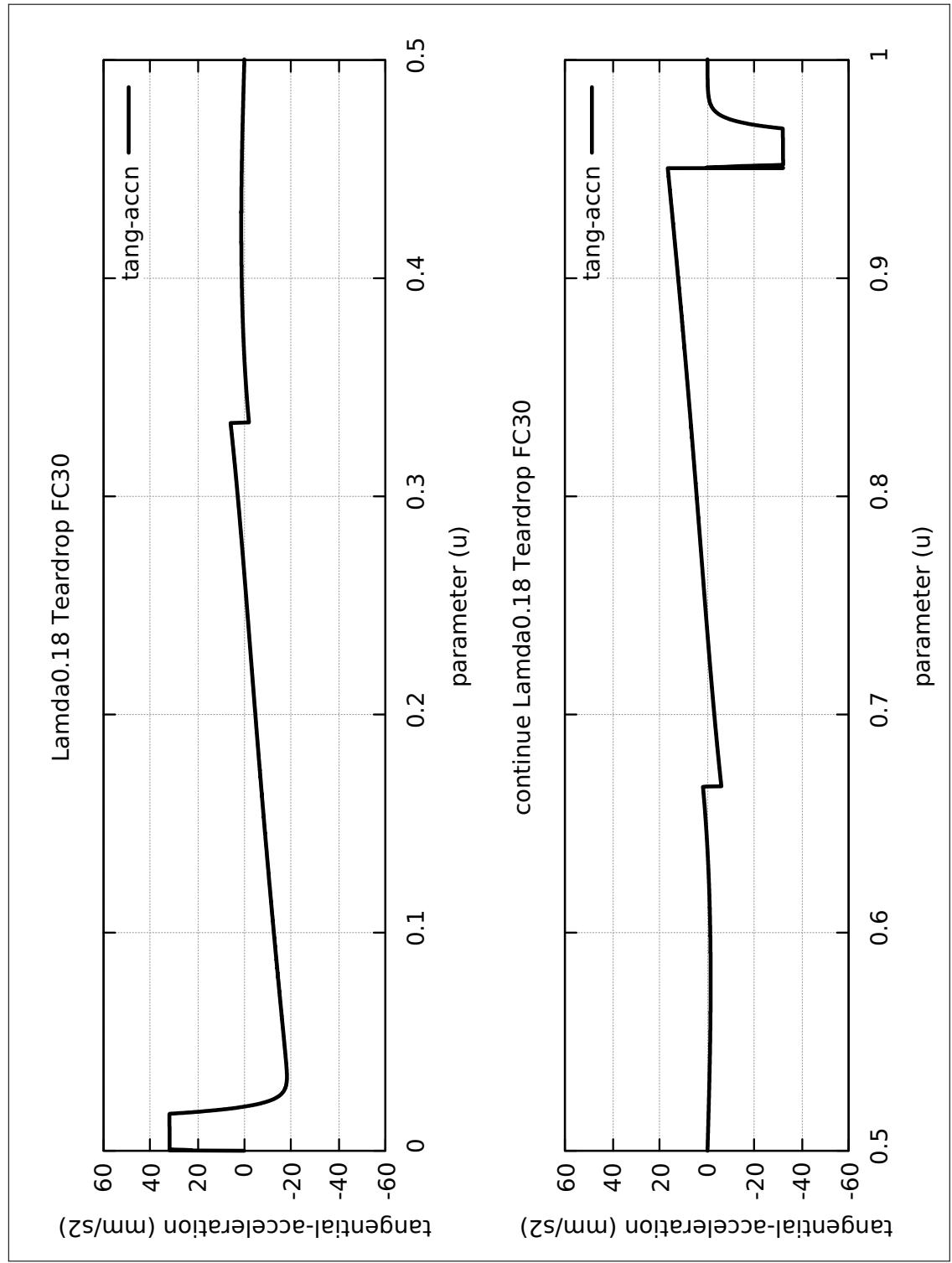
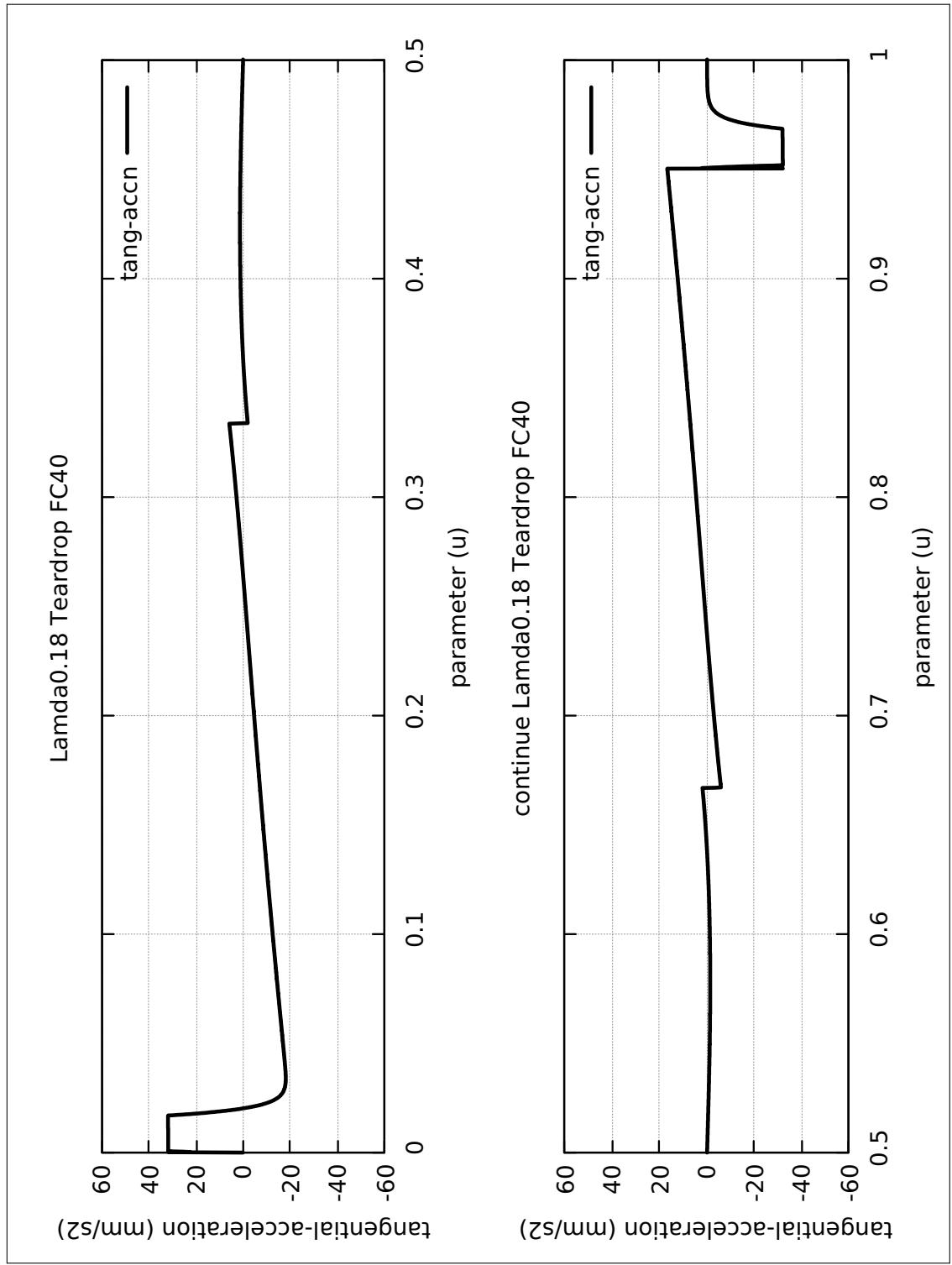


Figure 4.56: Teardrop FC40 Tangential Acceleration Run Profile



4.6 PERFORMANCE SUMMARY REST OF CURVES

The performance summary data of the interpolation algorithm for the rest of curves (9 different curves) are provided in the next nine(9) pages in landscape mode. The respective links are provided below.

4.6.1 Circle-Table Summary Performance data link [4.10]

4.6.2 Ellipse-Table Summary Performance data link [4.11]

4.6.3 Butterfly-Table Summary Performance data link [4.12]

4.6.4 Snailshell-Table Summary Performance data link [4.13]

4.6.5 Skewed-Astroid-Table Summary Performance data link [4.14]

4.6.6 Ribbon-10L-Table Summary Performance data link [4.15]

4.6.7 Ribbon-100L-Table Summary Performance data link [4.16]

4.6.8 AstEpi-Table Summary Performance data link [4.17]

4.6.9 SnaHyp-Table Summary Performance data link [4.18]

Table 4.10: Circle Table FC10-20-30-40 Run Performance data

1	Curve Type	CIRCLE		CIRCLE		CIRCLE	
2	User Feedrate Command FC(mm/s)	FC10		FC20		FC30	
3	User Lamda Acceleration Safety Factor	0.18		0.18		0.18	
4	Total Interpolated Points (TIP)	49641		24822		16549	
5	Total Sum-Chord-Error (SCE) (mm)	1.093913738449E-03		2.187639876000E-03		3.2811782287065E-03	
6	Ratio 1 = (SCE/TIP) = Chord-Error/Point	2.203694074232E-08		8.813665347893E-08		1.982824683989E-07	
7	Total Sum-Arc-Length (SAL) (mm)	4.963785816452E+02		4.963771594934E+02		4.963757335444E+02	
8	Total Sum-Chord-Length (SCL) (mm)	4.963775813150E+02		4.963771581682E+02		4.963757305630E+02	
9	Difference = (SAL - SCL) (mm)	3.302195636934E-07		1.325165840171E-06		2.981455850204E-06	
10	Percentage Difference = (SAL - SCL)/SAL	6.652574786747E-08		2.669675295946E-07		6.006449648363E-07	
11	Ratio 2 = (SCE/SCL) = Chord Error/Chord-Length	2.203789163406E-06		4.4077213023407E-06		6.610271383220E-06	
12	Total Sum-Arc-Theta (SAT) (rad)	6.283146611127E+00		6.283002050952E+00		6.282857458743E+00	
13	Total Sum-Arc-Area (SAA) (mm2)	5.235292684461E-05		2.093803820914E-04		4.710353817982E-04	
14	Ratio 3 = (SAA/SCL) = Arc-Area/Chord-Length	2.203789163406E-06		4.4077213023407E-06		6.610271383220E-06	
15	Average-Chord-Error (ACE) (mm)	2.203694074232E-08		8.813665347893E-08		1.982824683989E-07	
16	Average-Arc-Length (AAL) (mm)	9.999568526293E-03		1.999827402173E-02		2.999611636116E-02	
17	Average-Chord-Length (ACL) (mm)	9.999568519641E-03		1.999827396834E-02		2.999611618099E-02	
18	Average-Arc-Theta (AAT) (rad)	1.265742669445E-04		2.531325108155E-04		3.796747316136E-04	
19	Average-Arc-Area (AAA) (mm2)	1.054652031519E-09		8.435614281914E-09		2.846479222856E-08	
20	Algorithm actual runtime on computer (ART) (s)	17.277667419		8.160163431		5.399371536	
							4.053925000

Table 4.11: Ellipse Table FC10-20-30-40 Run Performance data

1	Curve Type	ELLIPSE	ELLIPSE	ELLIPSE
2	User Feedrate Command FC(mm/s)	FC10	FC20	FC40
3	User Lamda Acceleration Safety Factor	0.18	0.18	0.18
4	Total Interpolated Points (TIP)	21575	11296	8338
5	Total Sum-Chord-Error (SCE) (mm)	2.990951697698E-03	5.148661124856E-03	6.561982225601E-03
6	Ratio 1 = (SCE/TIP) = Chord-Error/Point	1.386368637108E-07	4.558354249540E-07	7.870915467915E-07
7	Total Sum-Arc-Length (SAL) (mm)	2.156436635306E+02	2.156499394203E+02	2.156478495089E+02
8	Total Sum-Chord-Length (SCL) (mm)	2.156436625529E+02	2.156499358521E+02	2.156478426167E+02
9	Difference = (SAL - SCL) (mm)	9.777115224097E-07	3.568242959773E-06	6.892209597709E-06
10	Percentage Difference = (SAL - SCL)/SAL	4.533921870933E-07	1.654645936542E-06	3.196048378596E-06
11	Ratio 2 = (SCE/SCL) = Chord Error/Chord-Length	1.386987988559E-05	2.387508767166E-05	3.042915776933E-05
12	Total Sum-Arc-Theta (SAT) (rad)	6.280515723261E+00	6.283169115421E+00	6.282291919271E+00
13	Total Sum-Arc-Area (SAA) (mm2)	5.185017589366E-05	1.275076043292E-04	1.849039783361E-04
14	Ratio 3 = (SAA/SCL) = Arc-Area/Chord-Length	1.386987988559E-05	2.387508767166E-05	3.042915776933E-05
15	Average-Chord-Error (ACE) (mm)	1.386368637108E-07	4.558354249540E-07	7.870915467915E-07
16	Average-Arc-Length (AAL) (mm)	9.995534603255E-03	1.909251345023E-02	2.586636074235E-02
17	Average-Chord-Length (ACL) (mm)	9.995534557936E-03	1.909251313431E-02	2.586635991565E-02
18	Average-Arc-Theta (AAT) (rad)	2.911150330611E-04	5.562788061462E-04	7.535434711852E-04
19	Average-Arc-Area (AAA) (mm2)	2.403364044390E-09	1.128885385827E-08	2.217871876407E-08
20	Algorithm actual runtime on computer (ART) (s)	7.13439876	7.633051722	11.041426063
				16.776571251

Table 4.12: Butterfly Table FC10-20-30-40 Run Performance data

1	Curve Type	BUTTERFLY	BUTTERFLY	BUTTERFLY
2	User Feedrate Command FC(mm/s)	FC10 0.18	FC20 0.18	FC40 0.18
3	User Lamda Acceleration Safety Factor			
4	Total Interpolated Points (TIP)	35656	18029	12343
5	Total Sum-Chord-Error (SCE) (mm)	1.938859834664E-03	3.534046223178E-03	4.846582536157E-03
6	Ratio 1 = (SCE/TIP) = Chord-Error/Point	5.437834342068E-08	1.960309642322E-07	3.926902071104E-07
7	Total Sum-Arc-Length (SAL) (mm)	3.560748506870E+02	3.560738974072E+02	3.560730284349E+02
8	Total Sum-Chord-Length (SCL) (mm)	3.560747025702E+02	3.560737108999E+02	3.560727930088E+02
9	Difference = (SAL - SCL) (mm)	1.481168304736E-04	1.865072539431E-04	2.354260789161E-04
10	Percentage Difference = (SAL - SCL)/SAL	4.159710526812E-05	5.237880543932E-05	6.611735799000E-05
11	Ratio 2 = (SCE/SCL) = Chord Error/Chord-Length	5.445092899523E-06	9.925041122094E-06	1.361121273884E-05
12	Total Sum-Arc-Theta (SAT) (rad)	2.212404302532E+01	2.212055105865E+01	2.211618559661E+01
13	Total Sum-Arc-Area (SAA) (mm2)	1.758667624644E-04	6.462621773264E-04	1.298932073590E-03
14	Ratio 3 = (SAA/SCL) = Arc-Area/Chord-Length	5.445092899523E-06	9.925041122094E-06	1.361121273884E-05
15	Average-Chord-Error (ACE) (mm)	5.437834342068E-08	1.960309642322E-07	3.926902071104E-07
16	Average-Arc-Length (AAL) (mm)	9.986673697574E-03	1.975115916392E-02	2.885051275603E-02
17	Average-Chord-Length (ACL) (mm)	9.986669543407E-03	1.975114881850E-02	2.885049368083E-02
18	Average-Arc-Theta (AAT) (rad)	6.205032400874E-04	1.227010819761E-03	1.791945032946E-03
19	Average-Arc-Area (AAA) (mm2)	4.932457228003E-09	3.584769122068E-08	1.052448609294E-07
20	Algorithm actual runtime on computer (ART) (s)	14.219218872	8.1882168602	8.667691811
				10.542664043

Table 4.13: Snailshell Table FC10-20-30-40 Run Performance data

1	Curve Type	SNAIL SHELL	SNAIL SHELL	SNAIL SHELL
2	User Feedrate Command FC(mm/s)	FC10	FC20	FC40
3	User Lamda Acceleration Safety Factor	0.18	0.18	0.18
4	Total Interpolated Points (TIP)	15621	9883	8370
5	Total Sum-Chord-Error (SCE) (mm)	5.115139395656E-03	6.269762299809E-03	6.764496555494E-03
6	Ratio 1 = (SCE/TIP) = Chord-Error/Point	3.274737129101E-07	6.344628921078E-07	8.082801476275E-07
7	Total Sum-Arc-Length (SAL) (mm)	1.385725614787E+02	1.385823558880E+02	1.385855208343E+02
8	Total Sum-Chord-Length (SCL) (mm)	1.385595406106E+02	1.385613905727E+02	1.385601641834E+02
9	Difference = (SAL - SCL) (mm)	1.302086811782E-02	2.096531529295E-02	2.535665098060E-02
10	Percentage Difference = (SAL - SCL)/SAL	9.396425943836E-03	1.512841599395E-02	1.829675338949E-02
11	Ratio 2 = (SCE/SCL) = Chord Error/Chord-Length	3.691654413053E-05	4.524898511695E-05	4.881992306637E-05
12	Total Sum-Arc-Theta (SAT) (rad)	1.894562150080E+01	1.8945335923343E+01	1.894325318416E+01
13	Total Sum-Arc-Area (SAA) (mm2)	1.070830851582E-04	3.068808285119E-04	5.164360050036E-04
14	Ratio 3 = (SAA/SCL) = Arc-Area/Chord-Length	3.691654413053E-05	4.524898511695E-05	4.881992306637E-05
15	Average-Chord-Error (ACE) (mm)	3.274737129101E-07	6.344628921078E-07	8.082801476275E-07
16	Average-Arc-Length (AAL) (mm)	8.871482809134E-03	1.402371543089E-02	1.655938831812E-02
17	Average-Chord-Length (ACL) (mm)	8.870649206822E-03	1.402159386488E-02	1.655635848768E-02
18	Average-Arc-Theta (AAT) (rad)	1.212907906581E-03	1.917156033539E-03	2.263502591010E-03
19	Average-Arc-Area (AAA) (mm2)	6.855511213715E-09	3.105452626107E-08	6.170820946393E-08
20	Algorithm actual runtime on computer (ART) (s)	17.308275283	24.17500989	29.295160546
				32.06253681

Table 4.14: Skewed-Astroid Table FC10-20-30-40 Run Performance data

1	Curve Type	SKEWED-ASTROID	SKEWED-ASTROID	SKEWED-ASTROID
2	User Feedrate Command FC(mm/s)	FC10	FC20	FC40
3	User Lamda Acceleration Safety Factor	0.18	0.18	0.18
4	Total Interpolated Points (TIP)	116194	58102	38738
5	Total Sum-Chord-Error (SCE) (mm)	5.163683043374E-04	1.032591177038E-03	1.548668518105E-03
6	Ratio 1 = (SCE/TIP) = Chord-Error/Point	4.444056908225E-09	1.777234775714E-08	3.997905150386E-08
7	Total Sum-Arc-Length (SAL) (mm)	1.161845691077E+03	1.161851349542E+03	1.161862568638E+03
8	Total Sum-Chord-Length (SCL) (mm)	4.4571428558819E+02	4.457142855374E+02	4.457142846207E+02
9	Difference = (SAL - SCL) (mm)	7.161314051951E+02	7.161370640044E+02	7.161426907744E+02
10	Percentage Difference = (SAL - SCL)/SAL	6.163739390653E+01	6.163758077028E+01	6.163776660469E+01
11	Ratio 2 = (SCE/SCL) = Chord Error/Chord-Length	1.158518631091E-06	2.316710975042E-06	3.474576811966E-06
12	Total Sum-Arc-Theta (SAT) (rad)	8.421542473789E+00	8.421464824576E+00	8.421387277982E+00
13	Total Sum-Arc-Area (SAA) (mm2)	4.343191363991E-05	1.736583480748E-04	3.905753943512E-04
14	Ratio 3 = (SAA/SCL) = Arc-Area/Chord-Length	1.158518631091E-06	2.316710975042E-06	3.474576811966E-06
15	Average-Chord-Error (ACE) (mm)	4.444056908225E-09	1.777234775714E-08	3.997905150386E-08
16	Average-Arc-Length (AAL) (mm)	9.999274406178E-03	1.999709728820E-02	2.999346814144E-02
17	Average-Chord-Length (ACL) (mm)	3.835982252648E-03	7.671370295474E-03	1.150616425177E-02
18	Average-Arc-Theta (AAT) (rad)	7.247891416685E-05	1.449452647042E-04	2.173990571800E-04
19	Average-Arc-Area (AAA) (mm2)	3.737911375032E-10	2.988904632878E-09	1.008274761471E-08
20	Algorithm actual runtime on computer (ART) (s)	38.427006969	19.305250287	12.910301226
				9.663656563

Table 4.15: Ribbon-10L Table FC10-20-30-40 Run Performance data

1	Curve Type	RIBBON-10L	RIBBON-10L	RIBBON-10L
2	User Feedrate Command FC(mm/s)	FC10 0.18	FC20 0.18	FC40 0.18
3	User Lamda Acceleration Safety Factor			
4	Total Interpolated Points (TIP)	7351	7352	7353
5	Total Sum-Chord-Error (SCE) (mm)	7.331686925442E-03	7.330650001960E-03	7.330843544266E-03
6	Ratio 1 = (SCE/TIP) = Chord-Error/Point	9.975084252302E-07	9.972316694273E-07	9.971223536814E-07
7	Total Sum-Arc-Length (SAL) (mm)	3.802699021307E+00	3.802672335504E+00	3.803478267930E+00
8	Total Sum-Chord-Length (SCL) (mm)	1.521079586306E+01	1.521068903483E+01	1.521191524702E+01
9	Difference = (SAL - SCL) (mm)	-1.140809684176E+01	-1.140801669932E+01	-1.140893646914E+01
10	Percentage Difference = (SAL - SCL)/SAL	-2.999999941577E+02	-2.999999919218E+02	-3.000000205782E+02
11	Ratio 2 = (SCE/SCL) = Chord Error/Chord-Length	4.820054776519E-04	4.819406921788E-04	4.818512589475E-04
12	Total Sum-Arc-Theta (SAT) (rad)	5.446619853644E+00	5.446616685362E+00	5.446717935311E+00
13	Total Sum-Arc-Area (SAA) (mm2)	1.338515095667E-06	1.338121185915E-06	1.338217831316E-06
14	Ratio 3 = (SAA/SCL) = Arc-Area/Chord-Length	4.820054776519E-04	4.819406921788E-04	4.818512589475E-04
15	Average-Chord-Error (ACE) (mm)	9.975084252302E-07	9.972316694273E-07	9.971223536814E-07
16	Average-Arc-Length (AAL) (mm)	5.173740165044E-04	5.173000048298E-04	5.173392638643E-04
17	Average-Chord-Length (ACL) (mm)	2.069496035791E-03	2.069199977531E-03	2.069357161916E-03
18	Average-Arc-Theta (AAT) (rad)	7.410367147815E-04	7.409354761749E-04	7.408484678062E-04
19	Average-Arc-Area (AAA) (mm2)	1.821108973696E-10	1.820325378744E-10	1.819805422838E-10
20	Algorithm actual runtime on computer (ART) (s)	65.683135834	59.539855092	75.452534184
				73.307374778

Table 4.16: Ribbon-100L Table FC10-20-30-40 Run Performance data

1	Curve Type	RIBBON-100L	RIBBON-100L	RIBBON-100L
2	User Feedrate Command FC(mm/s)	FC10 0.18	FC20 0.18	FC40 0.18
3	User Lamda Acceleration Safety Factor			
4	Total Interpolated Points (TIP)	7480	7348	7349
5	Total Sum-Chord-Error (SCE) (mm)	7.227413654822E-03	7.334488978808E-03	7.333764811636E-03
6	Ratio 1 = (SCE/TIP) = Chord-Error/Point	9.663609646773E-07	9.982971251951E-07	9.980627125254E-07
7	Total Sum-Arc-Length (SAL) (mm)	3.802433940498E+01	3.802572368417E+01	3.802757758292E+01
8	Total Sum-Chord-Length (SCL) (mm)	1.520973548479E+02	1.521028906780E+02	1.521393532475E+02
9	Difference = (SAL - SCL) (mm)	-1.140730154429E+02	-1.140771669938E+02	-1.141045169238E+02
10	Percentage Difference = (SAL - SCL)/SAL	-2.999999927099E+02	-2.999999893265E+02	-2.99999955199E+02
11	Ratio 2 = (SCE/SCL) = Chord Error/Chord-Length	4.751833890898E-05	4.822057586226E-05	4.821346349097E-05
12	Total Sum-Arc-Theta (SAT) (rad)	5.446630718238E+00	5.446603832206E+00	5.446627300213E+00
13	Total Sum-Arc-Area (SAA) (mm2)	1.265139232664E-04	1.339597112819E-04	1.339309239988E-04
14	Ratio 3 = (SAA/SCL) = Arc-Area/Chord-Length	4.751833890898E-05	4.822057586226E-05	4.821346349097E-05
15	Average-Chord-Error (ACE) (mm)	9.663609646773E-07	9.982971251951E-07	9.980627125254E-07
16	Average-Arc-Length (AAL) (mm)	5.084147533759E-03	5.175680370787E-03	5.175228304698E-03
17	Average-Chord-Length (ACL) (mm)	2.033658976439E-02	2.070272093072E-02	2.070091298694E-02
18	Average-Arc-Theta (AAT) (rad)	7.282565474312E-04	7.413371215743E-04	7.412394257230E-04
19	Average-Arc-Area (AAA) (mm2)	1.691588758743E-08	1.823325320292E-08	1.822685410980E-08
20	Algorithm actual runtime on computer (ART) (s)	26.306206937	37.694118593	42.698734839
				47.339581162

Table 4.17: AstEpi Table FC10-20-30-40 Run Performance data

1	Curve Type	ASTEP1	ASTEP1	ASTEP1
2	User Feedrate Command FC(mm/s)	FC10	FC20	FC40
3	User Lamda Acceleration Safety Factor	0.18	0.18	0.18
4	Total Interpolated Points (TIP)	76275	38169	25499
5	Total Sum-Chord-Error (SCE) (mm)	8.403732211685E-04	1.641842648547E-03	2.390002467065E-03
6	Ratio 1 = (SCE/TIP) = Chord-Error/Point	1.101782024240E-08	4.301620856599E-08	9.373293854673E-08
7	Total Sum-Arc-Length (SAL) (mm)	7.626471894183E+02	7.626564180797E+02	7.626597716718E+02
8	Total Sum-Chord-Length (SCL) (mm)	4.262622478423E+02	4.262622396899E+02	4.262622295515E+02
9	Difference = (SAL - SCL) (mm)	3.363849415760E+02	3.363941783897E+02	3.363975421203E+02
10	Percentage Difference = (SAL - SCL)/SAL	4.410754359858E+01	4.410822100426E+01	4.410846810274E+01
11	Ratio 2 = (SCE/SCL) = Chord Error/Chord-Length	1.971493430212E-06	3.851719659103E-06	5.606883043752E-06
12	Total Sum-Arc-Theta (SAT) (rad)	1.300410711943E+01	1.300415390419E+01	1.300401093322E+01
13	Total Sum-Arc-Area (SAA) (mm2)	2.898515736808E-04	6.811935608835E-04	9.367445601128E-04
14	Ratio 3 = (SAA/SCL) = Arc-Area/Chord-Length	1.971493430212E-06	3.851719659103E-06	5.606883043752E-06
15	Average-Chord-Error (ACE) (mm)	1.101782024240E-08	4.301620856599E-08	9.373293854673E-08
16	Average-Arc-Length (AAL) (mm)	9.998783195037E-03	1.998156618318E-02	2.991057226731E-02
17	Average-Chord-Length (ACL) (mm)	5.588565537959E-03	1.116805281099E-02	1.671747703944E-02
18	Average-Arc-Theta (AAT) (rad)	1.704920040831E-04	3.407082871566E-04	5.100012131627E-04
19	Average-Arc-Area (AAA) (mm2)	3.800136005465E-09	1.784724273956E-08	3.673796219754E-08
20	Algorithm actual runtime on computer (ART) (s)	27.836373672	14.372286925	9.138001576
				7.71877336

Table 4.18: SnaHyp Table FC10-20-30-40 Run Performance data

1	Curve Type	SNAHYP	SNAHYP	SNAHYP
2	User Feedrate Command FC(mm/s)	FC10	FC20	FC28**
3	User Lamda Acceleration Safety Factor	0.18	0.18	0.18
4	Total Interpolated Points (TIP)	38672	20223	16618
5	Total Sum-Chord-Error (SCE) (mm)	2.846873175820E-03	4.002975369043E-03	4.459254922025E-03
6	Ratio 1 = (SCE/TIP) = Chord-Error/Point	7.361778014067E-08	1.979515067275E-07	2.683549932012E-07
7	Total Sum-Arc-Length (SAL) (mm)	3.779474877648E+02	3.779592539969E+02	3.779667959094E+02
8	Total Sum-Chord-Length (SCL) (mm)	4.789870865777E+02	4.789986994127E+02	4.790063715425E+02
9	Difference = (SAL - SCL) (mm)	-1.010395988129E+02	-1.010394454158E+02	-1.010395756332E+02
10	Percentage Difference = (SAL - SCL)/SAL	-2.673376648446E+01	-2.673289365118E+01	-2.673223641934E+01
11	Ratio 2 = (SCE/SCL) = Chord Error/Chord-Length	5.943528031539E-06	8.356965006276E-06	9.309385400583E-06
12	Total Sum-Arc-Theta (SAT) (rad)	5.483921922253E+02	5.514331568816E+02	5.514225232921E+02
13	Total Sum-Arc-Area (SAA) (mm2)	2.706870610425E-01	2.760007501922E-01	2.777614243305E-01
14	Ratio 3 = (SAA/SCL) = Arc-Area/Chord-Length	5.943528031539E-06	8.356965006276E-06	9.309385400583E-06
15	Average-Chord-Error (ACE) (mm)	7.361778014067E-08	1.979515067275E-07	2.683549932012E-07
16	Average-Arc-Length (AAL) (mm)	9.773408698114E-03	1.869049817016E-02	2.274579020939E-02
17	Average-Chord-Length (ACL) (mm)	1.238620895704E-02	2.368700916886E-02	2.882628462072E-02
18	Average-Arc-Theta (AAT) (rad)	1.418096744913E-02	2.726897225208E-02	3.318424043402E-02
19	Average-Arc-Area (AAA) (mm2)	6.999742986798E-06	1.364853872971E-05	1.671549764280E-05
20	Algorithm actual runtime on computer (ART) (s)	18.694644926	16.525007035	16.49022576
				16.65385009

4.7 NOTABLE RESULTS REST OF CURVES

4.7.1 Algorithm validation and verification of Circle curve

Using PI = 3.14159_26535_89793_238 at (18-decimal digits in precision), the circumference of the Circle with radius R = 79.0 in this work is equal to 2*PI*R = 4.963716392672E+02. The total angle subtended by the circle in radians is 2*PI = 6.283185307180E+00. For comparisons, the results calculated by the algorithm extracted from Circle Table [4.10], in rows (7, 8, and 12), are provided in the table below.

Table 4.19: Circle - Algorithm Validation and Verification

CIRCUMFERENCE CIRCLE (mm)	
1	Mathematical Formula
SUM-ARC-LENGTH (SAL)	
2	SAL FC10-Circle-L0.18
3	SAL FC20-Circle-L0.18
4	SAL FC30-Circle-L0.18
5	SAL FC40-Circle-L0.18
SUM-CHORD-LENGTH (SAL)	
6	SCL FC10-Circle-L0.18
7	SCL FC20-Circle-L0.18
8	SCL FC30-Circle-L0.18
9	SCL FC40-Circle-L0.18
FULL ANGLE CIRCLE (rad)	
10	Mathematical Formula
SUM-ARC-THETA (SAT)	
11	SAT FC10-Circle-L0.18
12	SAT FC20-Circle-L0.18
13	SAT FC30-Circle-L0.18
14	SAT FC40-Circle-L0.18

From the results in the table above, it can be seen that values calculated by the algorithm are in very good agreement (up to 4 decimal places) with results calculated using mathematical formulas. It is said that the Circle curve validates and verifies the algorithm. In reverse, it can also be said that the algorithm validates and verifies the Circle curve. By definition, the sum-chord-lengths (SCL) must always be lower than the sum-arc-lengths (SAL). This fact is evident in the table above.

4.7.2 Algorithm validation and verification of Ellipse curve

For the Ellipse curve that is origin centered and elongated along the y-axis, the length of the semi-minor axis a , is the value of x when $y= 0.00$. Similarly, the length of the semi-major axis b , is the value of y when $x= 0.00$.

The perimeter of the Ellipse curve is calculated using the Ramanujan approximation formula shown in Appendix-Chap4 [2.1] with the lengths of semi-minor axis $a = 11.0$ and semi-major axis $b = 51.0$, respectively. The result for the Ellipse perimeter calculation is $2.156404170793E+02$. For a simple closed figure like the Ellipse, the total angle subtended (SAT) by the curve at the center in radians is $2*PI = 6.283185307180E+00$.

As comparisons, the results calculated by the algorithm extracted from Ellipse Table [4.11], in rows (7, 8, and 12), are provided in the table below.

Table 4.20: Ellipse - Algorithm Validation and Verification

PERIMETER ELLIPSE (mm)	
1 Mathematical Formula	2.156404170793E+02
SUM-ARC-LENGTH (SAL)	
2 SAL FC10-Ellipse-L0.18	2.156436635306E+02
3 SAL FC20-Ellipse-L0.18	2.156499394203E+02
4 SAL FC30-Ellipse-L0.18	2.156478495089E+02
5 SAL FC40-Ellipse-L0.18	2.156438658943E+02
SUM-CHORD-LENGTH (SAL)	
6 SCL FC10-Ellipse-L0.18	2.156436625529E+02
7 SCL FC20-Ellipse-L0.18	2.156499358521E+02
8 SCL FC30-Ellipse-L0.18	2.156478426167E+02
9 SCL FC40-Ellipse-L0.18	2.156438551446E+02
FULL ANGLE CIRCLE (rad)	
10 Mathematical Formula	6.283185307180E+00
SUM-ARC-THETA (SAT)	
11 SAT FC10-Ellipse-L0.18	6.280515723261E+00
12 SAT FC20-Ellipse-L0.18	6.283169115421E+00
13 SAT FC30-Ellipse-L0.18	6.282291919271E+00
14 SAT FC40-Ellipse-L0.18	6.280610715483E+00

From the results in the table above, it can be seen that values calculated by the algorithm are in very good agreement (up to 4 decimal places) with results calculated using mathematical formulas. It is said that the Ellipse curve validates and verifies the algorithm. In reverse, it can also be said that the algorithm validates and verifies the Ellipse curve. By definition, the sum-chord-lengths (SCL) must always be lower than the sum-arc-lengths (SAL). This fact is evident in the table above.

4.7.3 Case Sum-Arc-Length (SAL) less than Sum-Chord-Length (SCL)

The case where the Total Sum-Arc-Length (SAL) is less than Total Sum-Chord-Length (SCL) is abnormal and considered rogue. It should be the opposite or reverse. The values of SAL should always be greater than SCL.

The following table (in landscape mode) is an extract of data from the Teardrop Table [4.5], from rows (7, 8, 9 and 10), showing an abnormal case. Notice that the anomaly or abnormality happens only for FC30, the third row, and not for FC10, FC20 and FC40. The summary results of algorithm re-executions (as checks) of the Teardrop curve at FC10, FC20, FC30 and FC40 are provided in Listing [4.1] on the next page. There is a negative difference for (SAL-SCL) in FC30.

Figure [4.57], presents four(4) different curves the of Teardrop Differences (SAL-SCL), for FC10, FC20, FC30 and FC40, all running at a common Lamda = 0.18. However, only 3 curves are visible. The last curve is actually an overlap of FC30 and FC40 curves. The difference is so small that it is not visible at the current plot scale. The next Figure [4.58], for clarity shows FC30 and FC40 separation by shifting the left scale higher by 0.01 mm relative to the right scale.

The next Figure [4.59], shows both FC30 and FC40 on the same x and y-axes scales, displaying an apparent overlap. However, upon taking a closeup view as in Figure [4.60], of a specific region ($0.48 \leq u \leq 0.52$), and zooming in the y-axis, the separation becomes evident. The actual values for FC30 and FC40 are different, having a very small gap in the order of $1E - 5$, and both are positive.

The positiveness of both SAL and SCL throughout the full parameter range ($0.00 \leq u \leq 1.00$), can be seen through NAL (next-arc-length) and NCL (next-chord-length), in the next four(4) figures, namely Figure [4.61], [4.62], [4.63] and [4.64], for FC10, FC20, FC30 and FC40, respectively. The left and right y-axes scales have been shifted by 0.01

mm relative to each other, in order to separate NAL(u) and NCL(u), which otherwise shows an apparent overlap.

Since NAL(k) and NCL(k) are all positive throughout ($0.00 \leq k \leq 1.00$), which is the entire parametric curve, the Total SAL and Total NCL must also be both positive. Therefore, a negative difference between Total SAL and Total SCL, is not a problem. It just means that, for the particular case, the value of Total SCL calculated by the algorithm is higher than Total SAL. That is the explanation.

Why? Again it is about FC30 or Feedrate Command 30 mm/s that is having problems. And remember that the maximum machine limit for velocity is 30 mm/s for the x-axis direction, and also 30 mm/s for the y-axis direction. That means the algorithm is running at exactly the maximum allowable limits, and this may have led the approximate first-order arc length calculation into problems.

Notice that the same algorithm for the arc length calculation does not have problems for FC10, FC20 and FC40. These are feedrate commands that are out of the machine velocity limits. If FC30 is the threshold, above which the algorithm encounters negative difference problems, then it would not be correct for FC40, where there is no such negative difference. So FC30 is not a threshold point, and in this case it is just an anomaly.

The (SAL - SCL) negative difference anomalies are also found in algorithm executions for the Ribbon-10L link [4.15], Ribbon-100L link [4.16], and SnaHyp [4.18], curves. This just means the approximation equation used in calculating the arc-lengths is inferior (as reminded earlier) to the formula for calculating the chord-lengths.

Table 4.21: Anomaly at FC30 Teardrop Table FC10-20-30-40 Run Performance data

Run Type	Sum-Arc-Length Total SAL	Sum-Chord-Length Total SCL	Difference Total (SAL - SCL)	Percentage Difference 100*(SAL - SCL)/SAL
FC10-Teardrop-L018	1.018356741269E+02	1.018356732173E+02	9.095903408252E-07	8.931942058845E-07
FC20-Teardrop-L018	1.018418663504E+02	1.018418655699E+02	7.805327442156E-07	7.664163788301E-07
FC30-Teardrop-L018	1.018595636256E+02	1.018595666011E+02	-2.975420997586E-06	-2.921101261067E-06
FC40-Teardrop-L018	1.018355644559E+02	1.018355608386E+02	3.617250541765E-06	3.552050367760E-06

Listing 4.1: Anomaly Run FC30 Teardrop L=0.18 at FC30

FC10-L0.18-A28-TEARDROP (9) ALGORITHM EXECUTION STATISTICS	
xxxx	
FC10-L0.18-A28-TEARDROP SUM-ARC-LENGTH-(mm)	= 1.018356741269e+02
FC10-L0.18-A28-TEARDROP SUM-CHORD-LENGTH-(mm)	= 1.018356732173e+02
FC10-L0.18-A28-TEARDROP DIFF-ARC-LENGTH-CHORD-LENGTH-(mm)	= 9.095903408252e-07
FC10-L0.18-A28-TEARDROP PCNT-DIFF-ARC-CHORD-LENGTH	= 8.93194205845e-07
xxxx	
2023-10-19 22:25:06.592970712	Total program run duration = 4.597900838 seconds.
=====	=====
FC20-L0.18-A28-TEARDROP (9) ALGORITHM EXECUTION STATISTICS	
xxxx	
FC20-L0.18-A28-TEARDROP SUM-ARC-LENGTH-(mm)	= 1.018418663504e+02
FC20-L0.18-A28-TEARDROP SUM-CHORD-LENGTH-(mm)	= 1.018418655699e+02
FC20-L0.18-A28-TEARDROP DIFF-ARC-LENGTH-CHORD-LENGTH-(mm)	= 7.805327442156e-07
FC20-L0.18-A28-TEARDROP PCNT-DIFF-ARC-CHORD-LENGTH	= 7.664163788301e-07
xxxx	
2023-10-19 22:21:18.920618153	Total program run duration = 20.348336148 seconds.
=====	=====
FC30-L0.18-A28-TEARDROP (9) ALGORITHM EXECUTION STATISTICS	
xxxx	
FC30-L0.18-A28-TEARDROP SUM-ARC-LENGTH-(mm)	= 1.018595636256e+02
FC30-L0.18-A28-TEARDROP SUM-CHORD-LENGTH-(mm)	= 1.018595666011e+02
FC30-L0.18-A28-TEARDROP DIFF-ARC-LENGTH-CHORD-LENGTH-(mm)	= -2.975420997586e-06
FC30-L0.18-A28-TEARDROP PCNT-DIFF-ARC-CHORD-LENGTH	= -2.921101261067e-06
xxx	
2023-10-19 22:17:51.217462414	Total program run duration = 29.394313087 seconds.
=====	=====
FC40-L0.18-A28-TEARDROP (9) ALGORITHM EXECUTION STATISTICS	
xxxx	
FC40-L0.18-A28-TEARDROP SUM-ARC-LENGTH-(mm)	= 1.018355644559e+02
FC40-L0.18-A28-TEARDROP SUM-CHORD-LENGTH-(mm)	= 1.018355608386e+02
FC40-L0.18-A28-TEARDROP DIFF-ARC-LENGTH-CHORD-LENGTH-(mm)	= 3.617250541765e-06
FC40-L0.18-A28-TEARDROP PCNT-DIFF-ARC-CHORD-LENGTH	= 3.552050367760e-06
xxxx	
2023-10-19 22:28:14.569061988	Total program run duration = 34.387895739 seconds.

Figure 4.57: Plot of Teardrop FC10 FC20 FC30 and FC40 Differences SAL-SCL

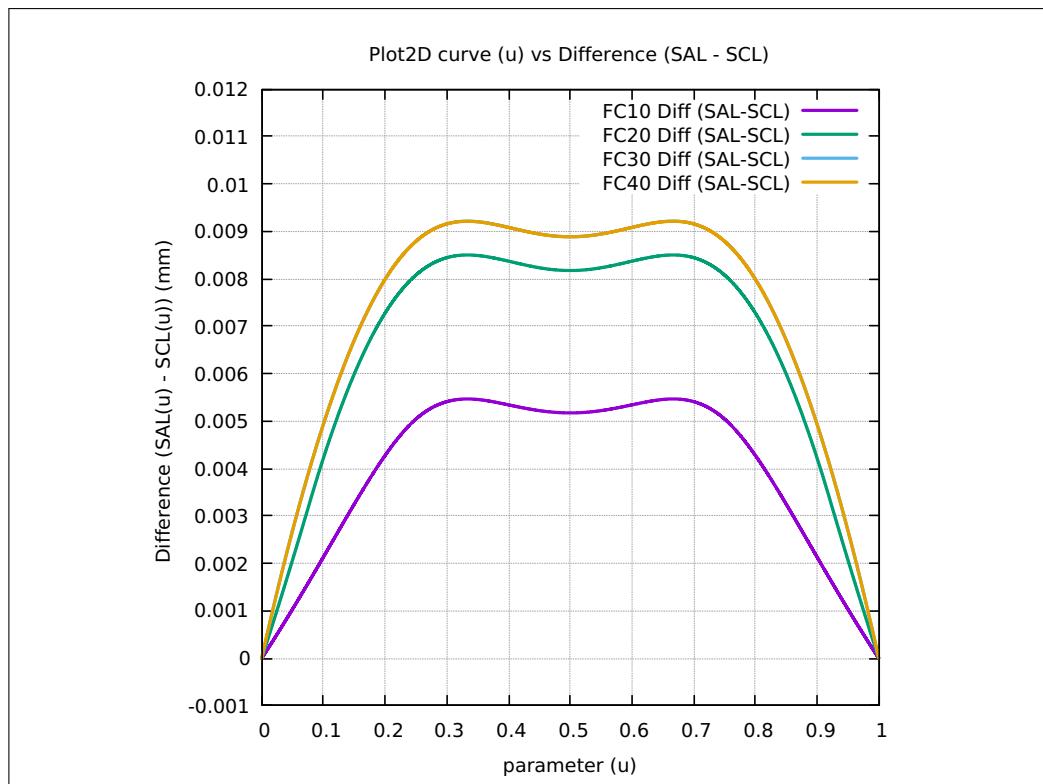


Figure 4.58: Teardrop Separation of FC30 and FC40 Difference SAL-SCL

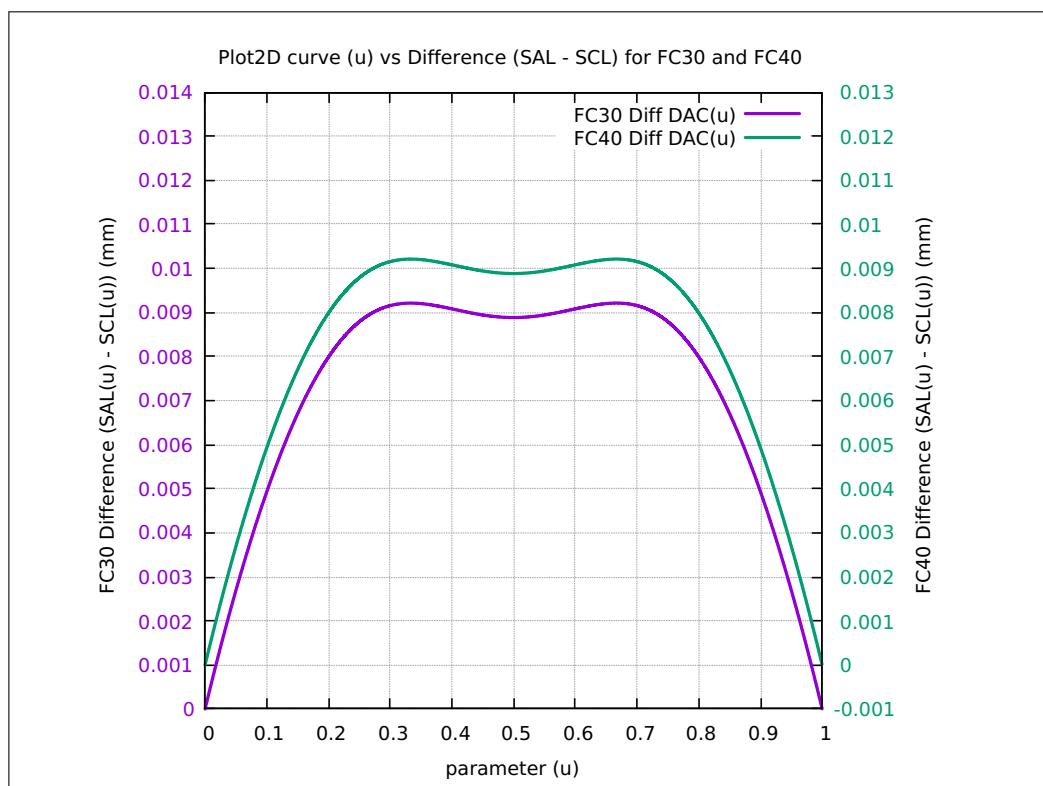


Figure 4.59: Plot of Teardrop Overlap FC30 and FC40 Differences SAL-SCL

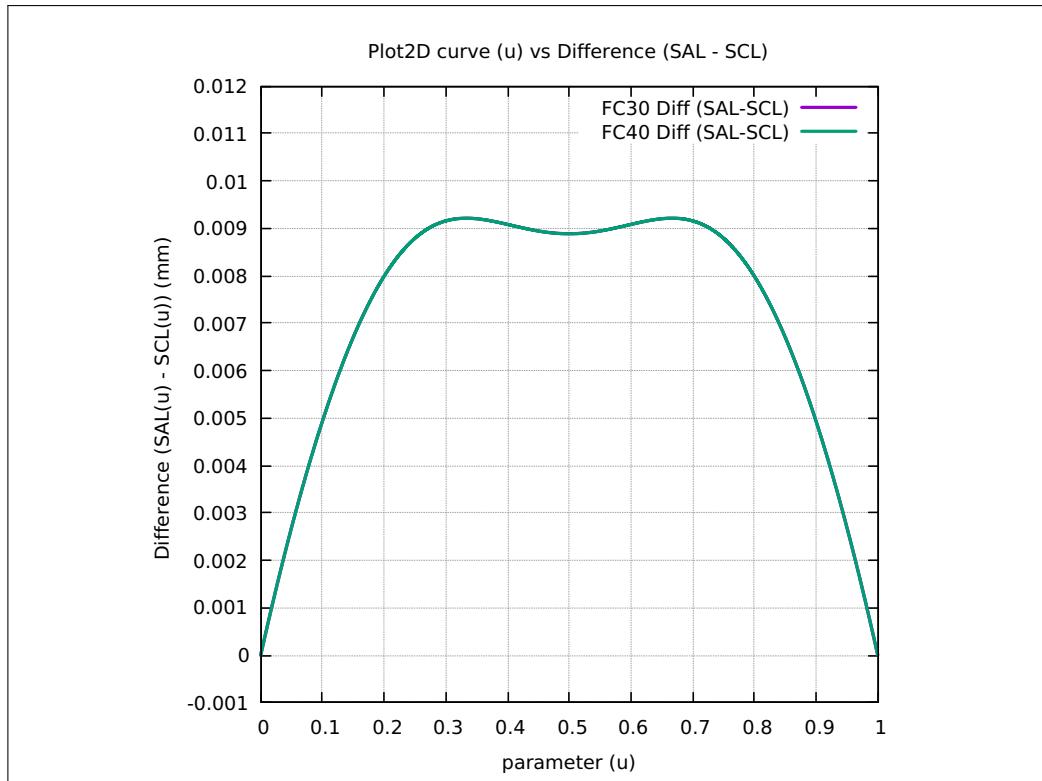


Figure 4.60: Teardrop Separation Closeup of FC30 and FC40 Difference SAL-SCL

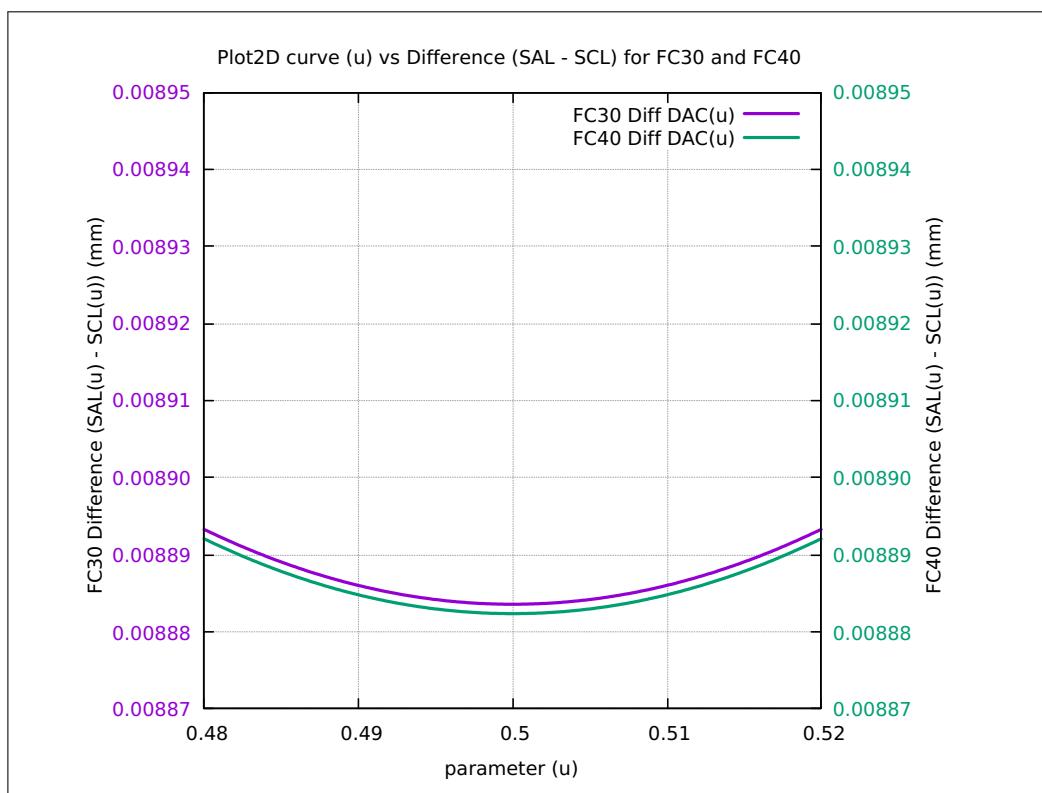


Figure 4.61: Plot of Teardrop FC10 NAL and NCL

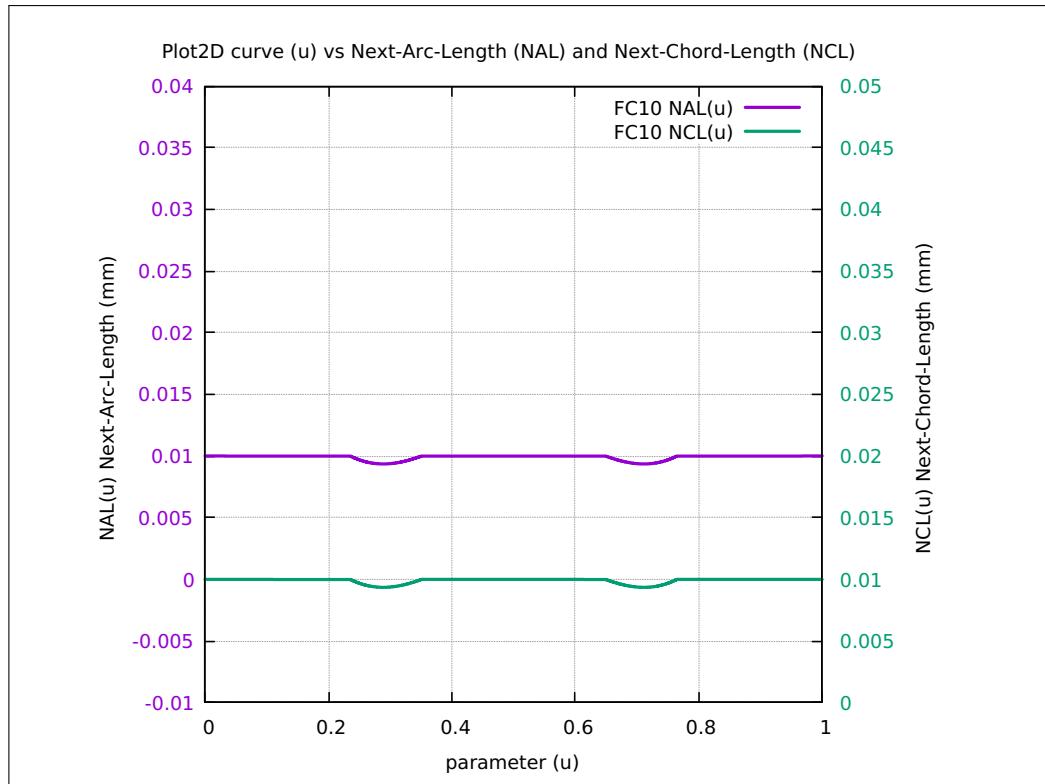


Figure 4.62: Plot of Teardrop FC20 NAL and NCL

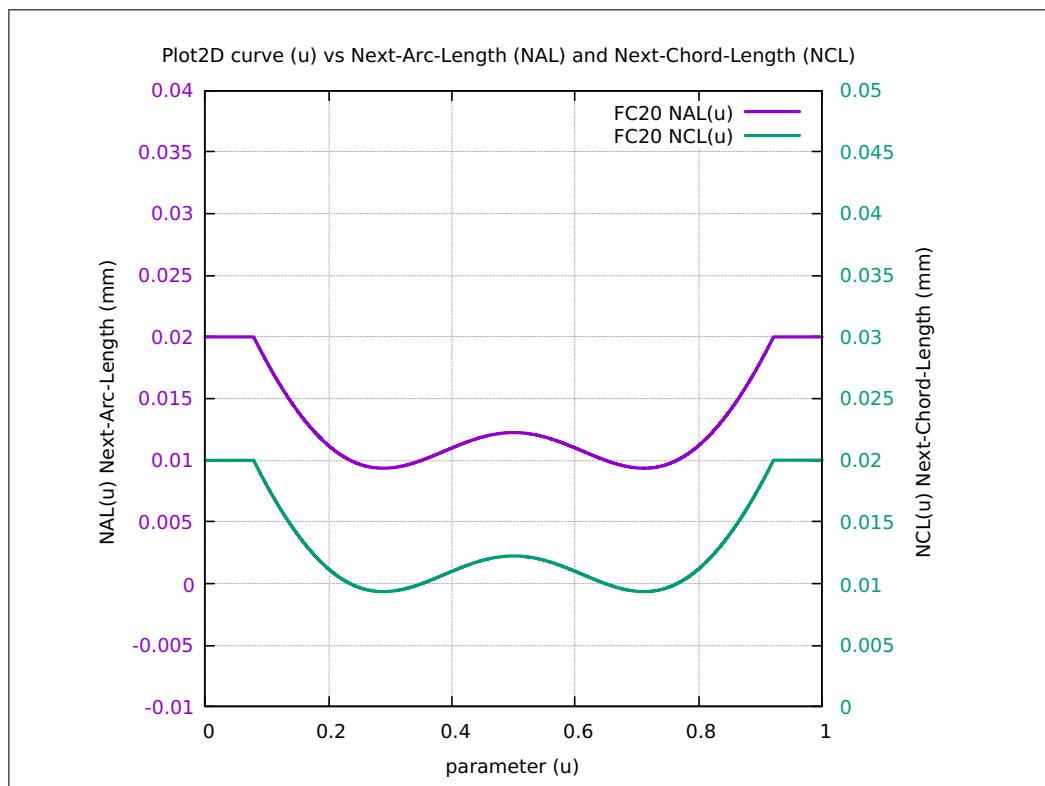


Figure 4.63: Plot of Teardrop FC30 NAL and NCL

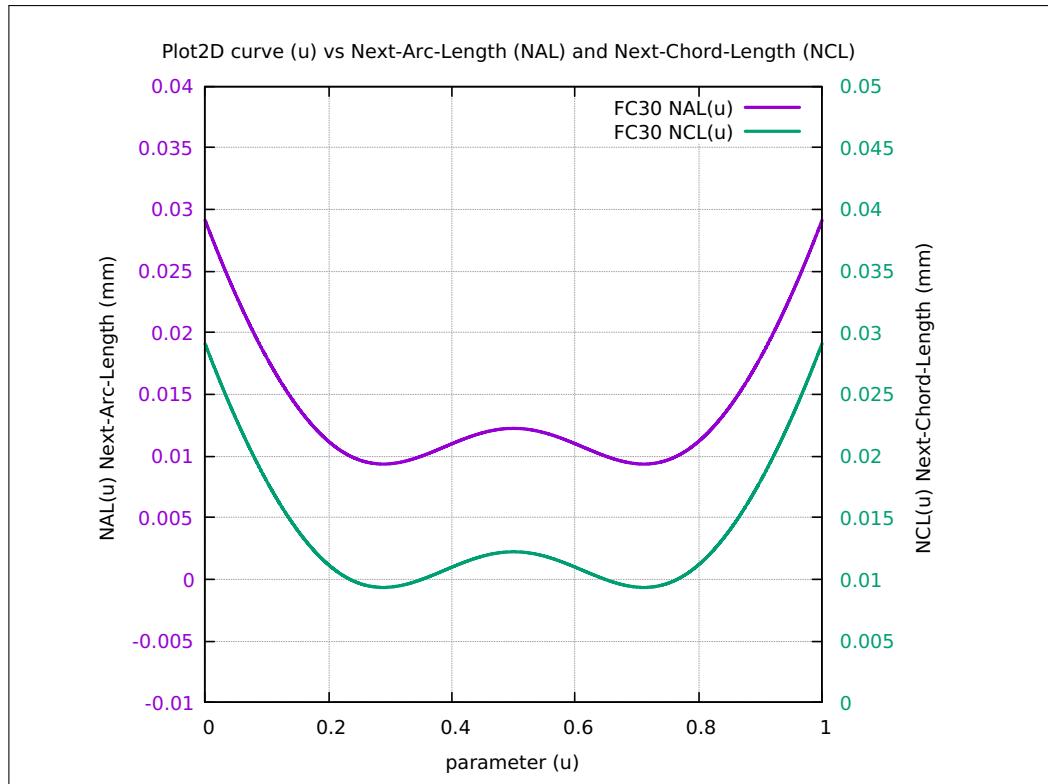
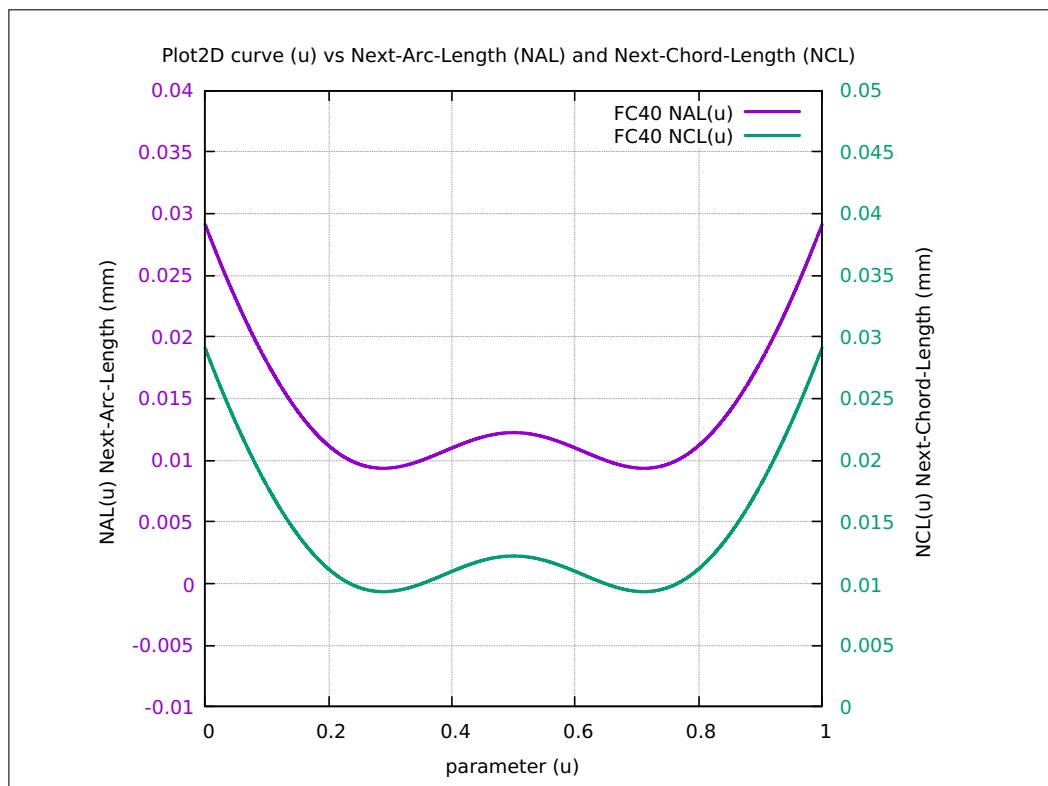


Figure 4.64: Plot of Teardrop FC40 NAL and NCL



4.7.4 SnaHyp curve algorithm machine epsilon problems

There is another notable case worthy of mention, where the algorithm developed in this work encountered machine epsilon problems. Among the ten(10) parametric curves selected in this work, this special case only occurs in the SnaHyp curve.

The machine epsilon is defined as the smallest floating-point (FP) number representable by a particular computer hardware. It only affects addition and subtraction operations in a computing device. Essentially, when numbers smaller than machine epsilon are added or subtracted from a bigger number, there is no change in the result of the addition or subtraction. The floating-point numbers smaller than machine epsilon, even though non-zero, are treated as exactly zero by the system. This is true only for addition and subtraction, but not for multiplication and division.

Among the four(4) algorithm executions at FC10, FC20, FC30 and FC40 for the SnaHyp curve, machine epsilon problems arise only for FC30 and FC40. There are no problems for algorithm runs at FC10 and FC20. Investigations carried out showed that the threshold or cut-off point for the onset of machine epsilon problems is at FC28 or 28 mm/s. Above this threshold Feedrate Command (FC), the algorithm encounters machine epsilon problems.

To see the machine epsilon run results, refer to Table [4.18], for the SnaHyp Summary Run Performance data FC10-FC20-FC30-FC40, located on a previous page. Due to the machine epsilon problem, the table results for FC30 and FC40 for the SnaHyp (columns) were replaced with FC25 and FC28, respectively.

A listing (landscape mode) of the actual algorithm execution results for the SnaHyp curve is provided on the next page in Listing [4.2], for the onset of the machine epsilon problem.

The run listing result shows that algorithm execution is successful for FC28 or 28 mm/s. However, it failed for FC29 or 29 mm/s. The failure is a result of five(5) consecutive times $u.next$ being zero, meaning there is no increase in parameter u . The finishing point for u is $u = 1.00$, whereas u stopped increasing at $u = 0.706801123$. The reason is $u.next$ already vanishes (value $u.next = 0.000000000$), which can be seen in the fourth column of the run data. Technically, $u.next$ is not really zero but a small number less than machine epsilon, which for an addition operation is considered zero by the system.

The algorithm developed, not only traps five(5) consecutive zero values then exits, but also traps conditions where variables that are supposed to vary (change in value, not stay constant, not necessarily zero) do vary, if not the algorithm also exits after five(5) consecutive cycles. In addition, the algorithm also traps conditions and exits, when output lines exceed a certain high value limit set by the user, to cater for unexpected or unknown errors. This will prevent the occurrence of infinite and unending loops in algorithm execution.

The machine epsilon problem is a fact of life, since floating point numeric values cannot be lower than machine epsilon values to make addition and subtraction operations valid. In this work the computation of machine epsilon value in C/C++ code is shown in a listing Appendix-Chap4 reference link at [App.2.2]. The machine epsilon values obtained are $2.2204460492503130808e - 16$ and $1.0842021724855044340e - 19$ for double and long double precision floating-point types, respectively.

The ranges of various floating-point and integer data types are provided in Appendix-Chap4 at reference link [App.2.3]. As shown in Appendix-Chap4 at reference link [App.2.4], only the operations of addition and subtraction are affected by machine epsilon.

A simple attempt to resolve the issue of machine epsilon is provided in Appendix-Chap4 at reference link [App.2.5]. The attempt by scaling up, perform the addition op-

eration, and then scale down the result, was unsuccessful. The real solution is in infinite precision arithmetic. This however, is beyond the scope of the current work in this thesis, especially just for the case of SnaHyp curve. Remember that the algorithm is already correct for FC10 and Fc20 for the SnaHyp curve.

Begin quote: *"In computer science, arbitrary-precision arithmetic, also called bignum arithmetic, multiple-precision arithmetic, or sometimes infinite-precision arithmetic, indicates that calculations are performed on numbers whose digits of precision are limited only by the available memory of the host system. This contrasts with the faster fixed-precision arithmetic found in most arithmetic logic unit (ALU) hardware, which typically offers between 8 and 64 bits of precision."* End quote. Source quoted from URL at [Arbitrary Precision Arithmetic].

Listing 4.2: SnaHyp Machine Epsilon Problems

```
(1) SNAHYP EXECUTION AT FC28 28 mm/ s (NO PROBLEM)
=====
FC28-L0.18-A28-SNAHYP (9) ALGORITHM EXECUTION STATISTICS

FC28-L0.18-A28-SNAHYP TOTAL-INTERPOLATED-POINTS          1.510200000000E+04
FC28-L0.18-A28-SNAHYP SUM-CHORD-ERROR-(mm)              4.700022719935E-03
FC28-L0.18-A28-SNAHYP SUM-CHORD-ERROR/TOT-INTERPOL-PNTS   3.112391709116E-07

FC28-L0.18-A28-SNAHYP SUM-ARC-LENGTH-(mm)                3.779678653378E+02
FC28-L0.18-A28-SNAHYP SUM-CHORD-LENGTH-(mm)              4.790071286890E+02
FC28-L0.18-A28-SNAHYP DIFF-ARC-LENGTH-CHORD-LENGTH-(mm) -1.010392633512E+02
FC28-L0.18-A28-SNAHYP PCNT-DIFF-ARC-CHORD-LENGTH        -2.673223641934E+01

FC28-L0.18-A28-SNAHYP SUM-CHORD-ERROR/SUM-CHORD-LENGTH   9.812009964860E-06

FC28-L0.18-A28-SNAHYP SUM-ARC-THETA-(rad)               5.481728747328E+02
FC28-L0.18-A28-SNAHYP SUM-ARC-AREA-(mm2)                 2.754368695310E-01

FC28-L0.18-A28-SNAHYP SUM-ARC-AREA/SUM-CHORD-LENGTH      9.812009964860E-06

FC28-L0.18-A28-SNAHYP AVG-CHORD-ERROR-(mm)              3.112391709116E-07
FC28-L0.18-A28-SNAHYP AVG-ARC-LENGTH-(mm)                2.502932688814E-02
FC28-L0.18-A28-SNAHYP AVG-CHORD-LENGTH-(mm)              3.172022572604E-02
FC28-L0.18-A28-SNAHYP AVG-ARC-THETA-(rad)                3.630043538393E-02
FC28-L0.18-A28-SNAHYP AVG-ARC-AREA-(mm2)                 1.823964436336E-05

2023-10-06 18:05:51    Total program run duration      16.653850090 seconds.
```

(2) SNAHYP EXECUTION AT FC29 29 mm/s (WITH MACHINE EPSILON PROBLEMS)

```
xxx
FC29-L0.18-A28-SNAHYP 11.942 0.706801123 0.000000000 0.001 4.149960e-03 2.285498e-14
FC29-L0.18-A28-SNAHYP 11.943 0.706801123 0.000000000 0.001 4.149960e-03 1.990422e-14
FC29-L0.18-A28-SNAHYP 11.945 0.706801123 0.000000000 0.001 4.149960e-03 1.496459e-14
FC29-L0.18-A28-SNAHYP 11.946 0.706801123 0.000000000 0.001 4.149960e-03 1.287859e-14
FC29-L0.18-A28-SNAHYP 11.947 0.706801123 0.000000000 0.001 4.149960e-03 1.110223e-14
```

```
NOTE: u_next_zero FIRST occurrence of (u_next < 1.0E-14)
NOTE: u_next_zero SECOND occurrence of (u_next < 1.0E-14)
NOTE: u_next_zero THIRD occurrence of (u_next < 1.0E-14)
NOTE: u_next_zero FOURTH occurrence of (u_next < 1.0E-14)
NOTE: u_next_zero FIFTH occurrence of (u_next < 1.0E-14)
```

RUN ERROR RECURSIVE NON STOP
 Five(5) consecutive occurrences of (u_next < 1.0E-14)
 Execute exit(1) at line 1310 in main().

STATUS: 0 NOT ACTIVATED, 1 ACTIVATED

EXIT ON u VALUE NOT CHANGING CONSECUTIVELY
 u_val_constant_1 0.0000000000E+00
 u_val_constant_2 0.0000000000E+00
 u_val_constant_3 0.0000000000E+00
 u_val_constant_4 0.0000000000E+00
 u_val_constant_5 0.0000000000E+00

EXIT ON u_next VALUE BECOMES ZERO CONSECUTIVELY
 u_next_zero_1 1.0000000000E+00
 u_next_zero_2 1.0000000000E+00
 u_next_zero_3 1.0000000000E+00
 u_next_zero_4 1.0000000000E+00
 u_next_zero_5 1.0000000000E+00

```
EXIT ON INTERPOLATED POINTS EXCEEDING 50,000 LINE LIMIT
display_line_limit_-03K_BELOW 1.000000000000E+00
display_line_limit_-03K 1.000000000000E+00
display_line_limit_-05K 1.000000000000E+00
display_line_limit_-10K 1.000000000000E+00
display_line_limit_-15K 0.000000000000E+00
display_line_limit_-20K 0.000000000000E+00
display_line_limit_-25K 0.000000000000E+00
display_line_limit_-30K 0.000000000000E+00
display_line_limit_-35K 0.000000000000E+00
display_line_limit_-40K 0.000000000000E+00
display_line_limit_-45K 0.000000000000E+00
display_line_limit_-50K 0.000000000000E+00
display_line_limit_-50K ABOVE 0.000000000000E+00
display_line_limit_-100K ABOVE 0.000000000000E+00
```

[2023-10-06 18:10:29] process exited with status 1, elapsed time: 08.51s
Alhamdulillah 3 times WRY.

4.8 OVERALL EXECUTION RESULTS

4.8.1 Feedrate Command (FC) and Lamda safety factor (LSF)

From this point onward, all algorithm executions will utilize Lamda = 0.18 and algorithm version Algo28. Execution runs for all ten(10) curves will be conducted for feedrate commands FC10, FC20, FC30 and FC40. However, the standard base feedrate command FC will be FC20 or 20 mm/s.

FC20 is considered the base case because it is reasonable and suitable among the four(4) feedrate commands. Running at FC10 or 10 mm/s is low compared to the machine velocity limit of 30 mm/s, that is, for both the X and Y motion axes. Running at FC40 or 40 mm/s is way above the machine velocity limit of 30 mm/s, which would be unsafe. Running at FC30 is essentially running at exactly the machine velocity limit. Since running at FC10 means not fully utilizing the machine, therefore, F20 was chosen as the base case because most people would want to run their machine safely, including taking full advantage of its capabilities.

The idea of running at FC30 and FC40 is to test the robustness of the algorithm developed in this work, that is, when applied against parametric curves of various characteristics and sizes. This is also to confirm that one single algorithm can satisfy all the parametric curves without fail.

In this work, when there is no mention of feedrate command, it is taken nominally as FC20. Similarly, when there is no mention of lamda safety factor, it is taken as lamda = 0.18. All calculations and reporting presented in this work are based on the latest algorithm version 28.

4.8.2 Algorithm Validation and Verification (V & V)

Technically, both validation and verification must be considered in this work. The term is called software V & V, in software engineering vocabulary. Essentially, validation is about doing the correct things, while verification is about doing the correct things correctly.

As examples, there are very strong reasons why the trivial circle and ellipse curves were chosen in this work. These two geometrical objects have well known exact mathematical formulae for calculation of circumference or perimeter. In addition, for a circle and an ellipse that are both centered at the origin, the total sum of the subtended angle swept by the arcs to the origin, for both curves is 2π radians.

Therefore, having both calculations for the circumference and total sum subtended angle in the algorithm are validation procedures. This is doing the correct things. While getting those calculations to give the correct answers is verification, or doing the correct things correctly.

The formula for the circumference of a circle is $C = 2 * \pi * R$, where R is the radius of the circle, while the formula for the perimeter of an ellipse is given by the Ramanujan approximation in Appendix-Chap4 [2.1], respectively.

For the circle and ellipse curves, the algorithm validation and verification results for this work have already been provided in the previous sections, for Circle Ref [4.7.1], and for Ellipse Ref [4.7.2], respectively.

There is another aspect of algorithm validation, that is, when the algorithm executes it must display (or exhibit behavior) that it executes correctly.

For an algorithm in this work that calculates the running feedrate (velocity) from point to point, it must be shown visually that the motions follow the calculated feedrate motions. For example, when the feedrate increases the machine feedrate must also increase correspondingly. When the algorithm calculation shows a corner coming in its interpolated path, the machine must also turn at this corner, accordingly. When the algorithm says that the feedrate at the corner shall be so and so, the machine must also be seen as obeying such instructions. This is visual validation.

For physical verification, for example, in the part (or product) generated by the machine cutting a circle of radius so and so, measurement of the radius of that circle using a micrometer is verification. Similarly, measuring the lengths of semi-major and semi-minor axes of a cut ellipse-shaped product using a micrometer is considered verification.

4.8.3 LinuxCNC-Axis Simulation Run Validation (SRV)

Since the algorithm also generates a G-Code file, the validation of resulting G-code files is easily conducted. This can be done by executing the G-code files on the LinuxCNC-Axis software, a free, and open-source Linux software to control CNC machines.

The first visual validation is that, the correct curve is displayed correctly in the software, that is, in terms of shapes and dimensions. For example, a circle must be a closed figure. Having the resulting circle that does not meet its closing "end-point", means the interpolation have gone wrong making the curve way off its correct path. This can be seen through a live preview in the GUI panel of LinuxCNC. In fact, it is known among CNC practitioners, that cutting a circle is the first basic act of validation.

In this work, simulation runs were conducted to ensure that the G-codes will be driven by CNC machine correctly, in terms of feedrate and curve path tracing for all the ten(10) selected curves. These simulations were successful. For all curves, the machine feedrate is smooth throughout as the CNC pen tool (simulating the laser cutter) moves while it traces the curve trajectory.

The LinuxCNC-Axis curve validations are shown in the figures listed below.

1. Figure [5] for the Circle curve
2. Figure [6] for the Ellipse curve
3. Figure [7] for the Teardrop curve
4. Figure [8] for the Butterfly curve
5. Figure [9] for the Snailshell curve
6. Figure [10] for the Skewed-Astroid curve
7. Figure [11] for the Ribbon-10L curve
8. Figure [12] for the Ribbon-100L curve
9. Figure [13] for the AstEpi curve
10. Figure [14] for the SnaHyp curve

4.8.4 LinuxCNC-Axis Real Run Validation (RRV)

In addition, in this work real runs on the CNC machine were also executed where the LinuxCNC internal engine drive G-code interpreted signals through the standard parallel port of the computer to the CNC electrical motors. LinuxCNC supports the IEEE 1884 parallel port as the hardware driver port. The point-to-point move of the CNC pen tool can be observed by the line-to-line changes on the LinuxCNX GUI display panel.

As a monitoring measure on validation, the G-code that this author wrote for each line, includes the current parameter u value, chord error (epsilon), current feedrate-limit, and the difference between the current feedrate-limit and the current running-feedrate. This is not a standard practice in the NGC G-code industry. This is a validation proof where each line shows that simultaneous constraints on feedrate and chord-error were fully satisfied.

Since the LinuxCNC software execution was conducted on a realtime operating system, at Ref [.1.2], (RTOS, with known, stable and acceptable latency), this makes the CNC machine execution realtime. Note that the LinuxCNC software does not physically run on a non-realtime general purpose operating system (GPOS). In GPOS, LinuxCNC only allows simulations but not drive the actual electrical signals to the CNC machine.

On the next pages, in the figure for the LinuxCNC-Axis panel, the lower panel displays a line cursor that shows the current line being executed. The letter F stands for running feedrate (vel), X and Y for the X and Y axis positions of the cutter tool (pen). These are the three(3) standard G-Code commands. The information for each line within parenthesis (# xxxxxx #) are G-code comments, containing the author's additional information for each line obtained from the algorithm. The information specify the current parameter u, the chord-error eps, the current feedrate, the x-axis and y-axis feedrate, the feedrate limit, and the difference between the feedrate limit and the current running feedrate. The current Feedrate Command (FC) is at the end of the line.

4.8.5 Overall Total Interpolated Points (TIP)

The histogram for the total number of interpolated points for all ten(10) curves in this work is provided in Figure [4.65] on the next page. The data table corresponding to the histogram is provided in Table [4.22].

With the exception of the Ribbon-10L and Ribbon-100L curves, it can be seen generally the trend, that as the feedrate command (FC) increases the total number of interpolated points decreases. This is expected since an increase in feedrate will increase the chord-length, thus reducing the total number of interpolated points for the same length of the curve.

The special case and peculiarity of the the Ribbon-10L and Ribbon-100L curves will be explained and discussed in sections Ribbon-10L [App.9] and Ribbon-100L [App.10], respectively. The effect is pure stretching the chord-lengths in Ribbon-100L to ten times that of Ribbon-10L, without increasing the total number of interpolated points.

Notice that the highest number of interpolated points falls to the Skewed-Astroid curve. This is the concave shaped diamond curve with two sharp cusps on the vertical axis and two broad cusps on the horizontal axis. These four(4) cusps generate most of the interpolated points.

From the histogram and data table, it is important to realize that the single algorithm developed in this work managed to successfully run all four(4) values of feedrate commands against all of the ten(10) curves. However, at row 10 in the data table, SnaHyp curve, the last two column entries for this curve are runs for FC25 and FC28, meaning not FC30 and FC40, respectively. This is due to the fact that at FC29, the parameter u does not move further after $u = 0.706$. At this point the increment $u\text{-next}$ vanishes, that is, $u\text{-next} = 0.000000$. The run did not complete. Run completion means u reaches 1.00000. This issue has already been discussed in section SnaHyp [4.7.4].

Figure 4.65: Histogram Overall Total Interpolated Points TIP

Overall Total Interpolated Points TIP for each curve
at Feedrate Commands FC10, FC20, FC30 and FC40
with all lambda = 0.18

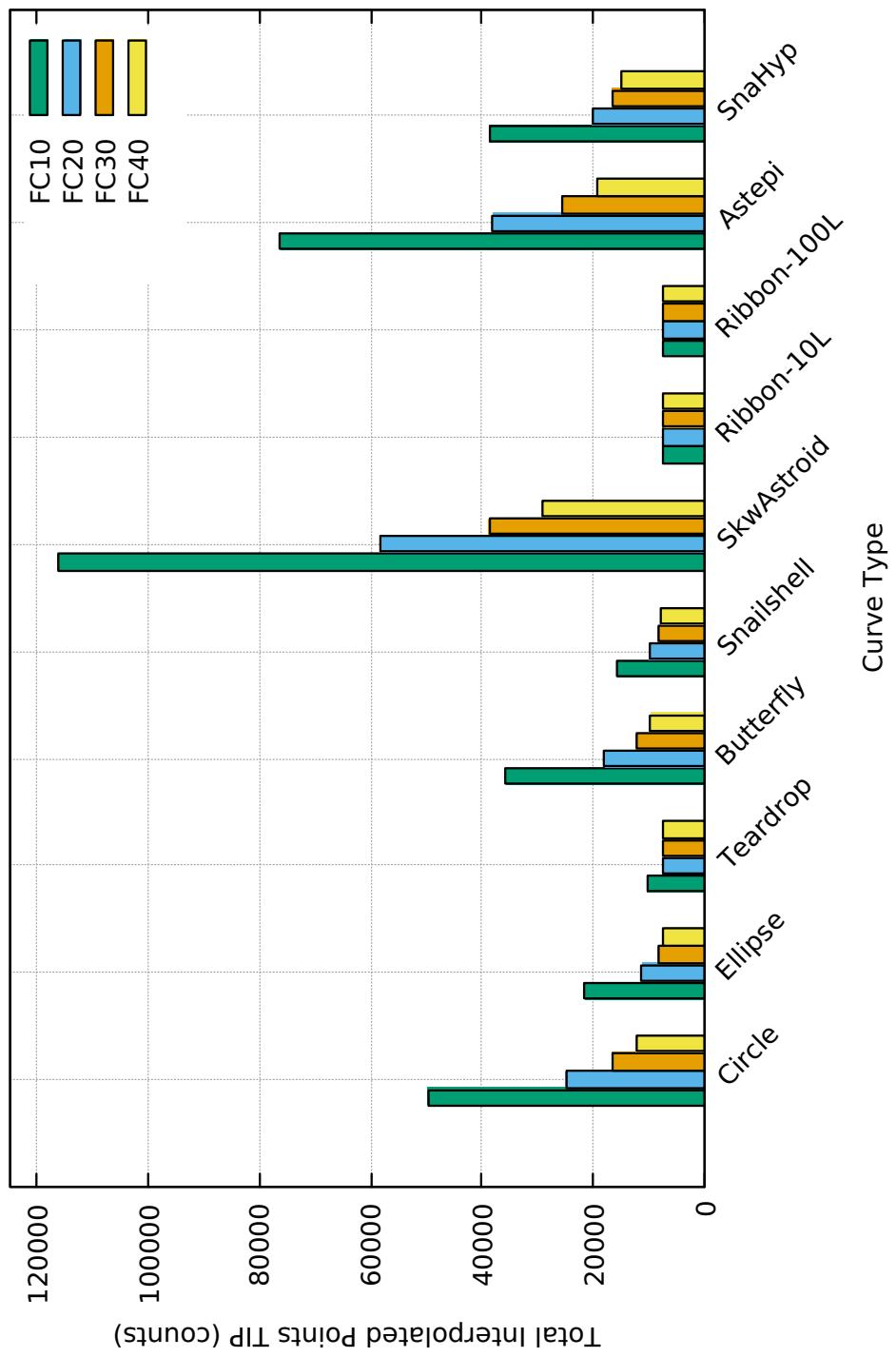


Table 4.22: Overall Total Interpolated Points Data TIP

OVERALL INTERPOLATED POINTS (TIP)				Total Interpolated points	Total Interpolated points	Total Interpolated points
	Description	Lambda Acceleration Safety Factor	Feedrate Command (mm/s)	0.18	0.18	0.18
	Curve Type			FC10	FC20	FC30
1	Circle			49641	24822	16549
2	Ellipse			21575	11296	8338
3	Teardrop			10261	7599	7347
4	Butterfly			35656	18029	12343
5	Snailshell			15621	9883	8370
5	Skewed-Astroid			116194	58102	38738
7	Ribbon-10L			7351	7352	7353
8	Ribbon-100L			7480	7348	7349
9	AstEpi			76275	38169	25499
10	SnaHyp – FC10 FC20 FC25 FC28			38672	20223	16618

4.8.6 Overall Total Sum-Arc-Length (SAL)

The histogram for the total Sum-Arc-Length (SAL) for all ten(10) curves in this work is provided in Figure [4.66] on the next page. The data table corresponding to the histogram is provided in Table [4.23].

Notice that the results for SAL of each curve is a flat top for the four feedrate commands FC10, FC20, FC30 and FC40. This is expected since the physical lengths of each curve is a static constant. The total Sum-Arc-Length (SAL) is really the length of the curve.

The flat top for each curve means the algorithm is consistent in calculating SAL, even though the calculation itself is a first-order approximation. The length of a curve must be the same physically, irrespective of whether the algorithm calculates it at FC10, FC20, FC30 or FC40.

Figure 4.66: Histogram Overall Total Sum-Arc-Length SAL

Overall Total Sum-Arc-Length SAL for each curve
at Feedrate Commands FC10, FC20, FC30 and FC40
with all $\lambda = 0.18$

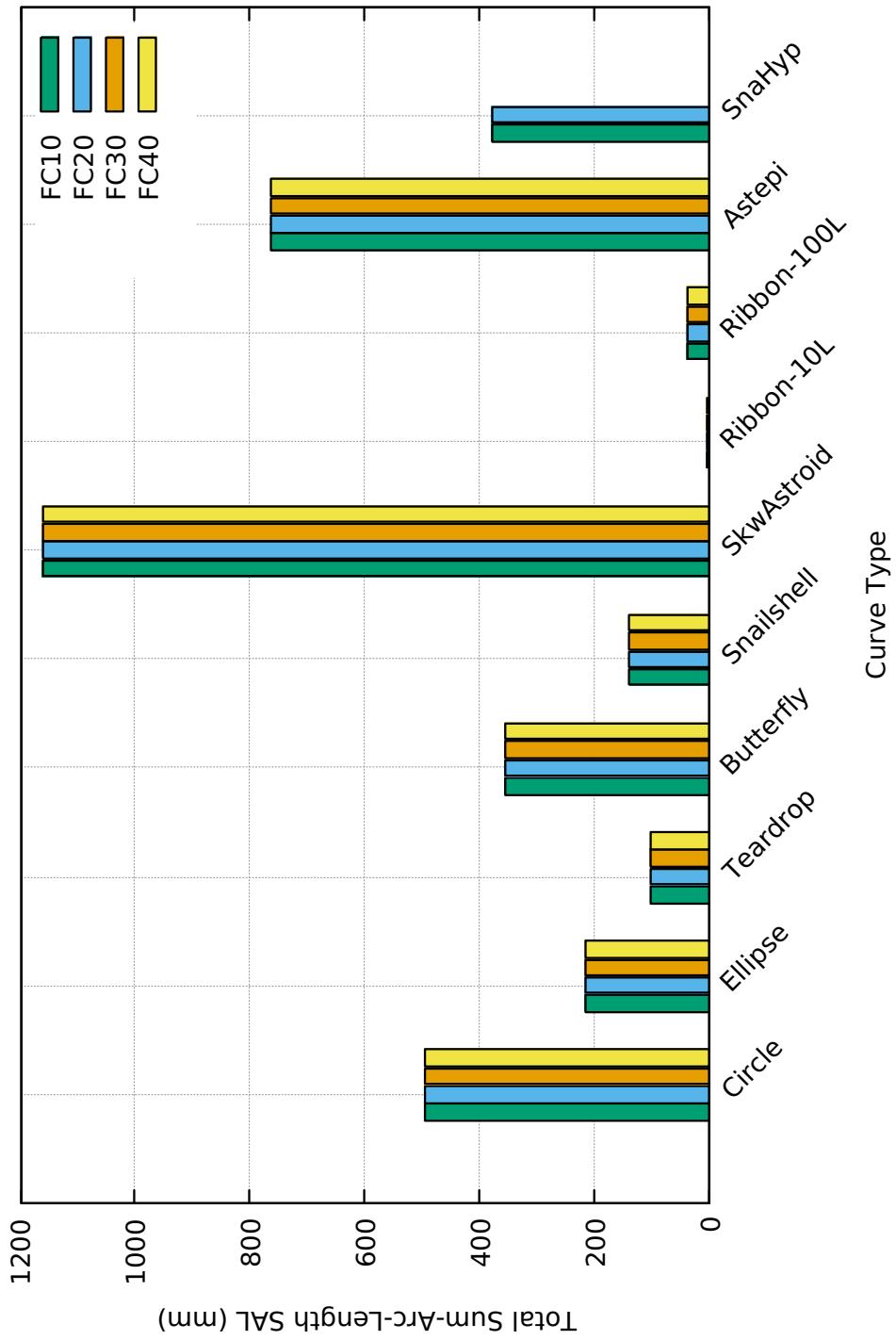


Table 4.23: Overall Total Sum-Arc-Length Data SAL

OVERALL SUM-ARC-LENGTH (SAL)			
	Description	Sum-Arc-Length	Sum-Arc-Length
	Lambda Acceleration Safety Factor	0.18	0.18
	Feedrate Command (mm/s)	FC10	FC20
	Curve Type		
1	Circle	4.963785816452E+02	4.963771594934E+02
2	Ellipse	2.156436635306E+02	2.156499394203E+02
3	Teardrop	1.018356741269E+02	1.018418663504E+02
4	Butterfly	3.560748506870E+02	3.560738974072E+02
5	Snailshell	1.385725614787E+02	1.385823558880E+02
5	Skewed-Astroid	1.161851349542E+03	1.161856975395E+03
7	Ribbon-10L	3.802699021307E+00	3.802672335504E+00
8	Ribbon-100L	3.802433940498E+01	3.802572368417E+01
9	AstEpi	7.626471894183E+02	7.626564180797E+02
10	SnaHyp	3.779474877648E+02	3.779592539969E+02

4.8.7 Overall Total Sum-Chord-Length (SCL)

The histogram for the total Sum-Chord-Length (SCL) for all ten(10) curves in this work is provided in Figure [4.67] on the next page. The data table corresponding to the histogram is provided in Table [4.24].

While the total Sum-Arc-Length (SAL) is the sum of all arc segments for the curve, the total Sum-Chord-Length (SCL) is the sum of all linear chords spanning the curve. It is expected that the total SAL is always greater than the total SCL.

Note that the total SCL calculation is exact because each chord-length is a linear length from point to point between two (x, y) coordinate points. On the other hand, the total SAL calculation is a first-order approximate arc calculation.

From the histogram, the flat top for each curve means the algorithm is consistent in calculating SCL. Even though the chord-lengths calculated by the algorithm at different values of feedrate command FC10, FC20, FC30 or FC40 vary (longer for higher FC), their total sums are the same. This fact verifies that the algorithm is correct.

Figure 4.67: Histogram Overall Total Sum-Chord-Length SCL

Overall Total Sum-Chord-Length SCL for each curve
at Feedrate Commands FC10, FC20, FC30 and FC40
with all lambda = 0.18

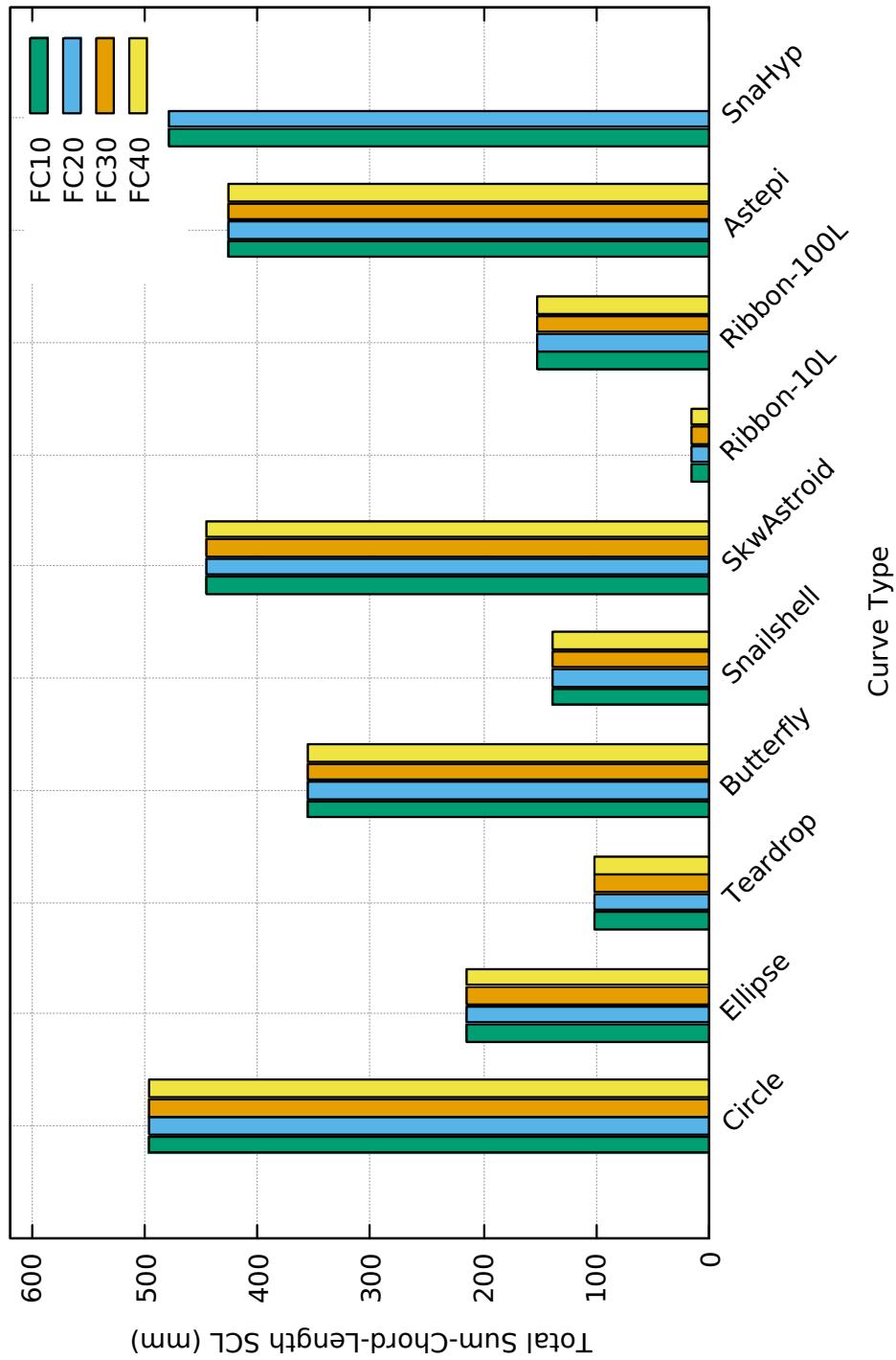


Table 4.24: Overall Total Sum-Chord-Length SCL

OVERALL SUM-CHORD-LENGTH (SCL)			
	Description	Sum-Chord-Length	Sum-Chord-Length
	Lambda Acceleration Safety Factor	0.18	0.18
	Feedrate Command (mm/s)	FC10	FC20
	Curve Type		
1	Circle	4.963785813150E+02	4.963771581682E+02
2	Ellipse	2.156436625529E+02	2.156499358521E+02
3	Teardrop	1.018356732173E+02	1.018418655699E+02
4	Butterfly	3.560747025702E+02	3.560737108999E+02
5	Snailshell	1.385595406106E+02	1.385613905727E+02
5	Skewed-Astroid	4.457142855374E+02	4.457142846207E+02
7	Ribbon-10L	1.521079586306E+01	1.521068903483E+01
8	Ribbon-100L	1.520973548479E+02	1.521028906780E+02
9	AstEpi	4.262622478423E+02	4.262622396899E+02
10	SnaHyp	4.789870865777E+02	4.78986994127E+02
		0.000000000000E+00	0.000000000000E+00

4.8.8 Overall Percentage Difference between SAL and SCL

The histogram for the difference between Sum-Arc-Length (SAL) and Sum-Chord-Length (SCL) for all ten(10) curves in this work is provided in Figure [4.68] and Figure [4.69] on the next 2 pages. For the first figure, the Y-scale is linear, while in the second figure the y-axis is log base 10. The data table corresponding to the histogram is provided in Table [4.25].

The main reason for having this metric as a percentage is to gauge size of the difference (SAL-SCL) relative to itself, since different curves have different total lengths.

In the linear scale, the histogram for five(5) curves comprising the Circle, Ellipse, Teardrop, Butterfly and Snailshell curves are not visible due to their very small values. Two(2) curves, the Skewed-Astroid and AstEpi curves have large positive percentage differences, while the other three(3) curves, the Ribbon-10L, Ribbon-100L and the SnaHyp curves have large negative percentage differences.

In the log base 10 scale, the histogram for five(5) curves comprising the Circle, Ellipse, Teardrop, Butterfly and Snailshell curves are now visible. Their values for the percentage difference of (SAL-SCL) is less than 1 percent. This means that the first-order approximation calculation for the arc-length is good for these five(5) curves, but performed badly for the rest of the curves. In this log scale, you may notice that FC30 (orange) for the Teardrop curve is missing. It is not missing but has a small negative value. This particular issue has been covered in a previous section with link [4.7.3].

Since calculation for the chord-length is exact, the data in this section establishes the fact that approximate calculations of the arc-length is not reliable for the later five(5), comprising the Skewed-Astroid, Ribbon-10L, Ribbon-100L, AstEpi and SnaHyp curves.

Figure 4.68: Histogram Lineal Overall Percentage Difference between SAL and SCL

Overall Percentage Difference $100 * (\text{SAL-SCL}) / \text{SAL}$ Linear Scale for each curve
 at Feedrate Commands FC10, FC20, FC30 and FC40
 with all lambda = 0.18

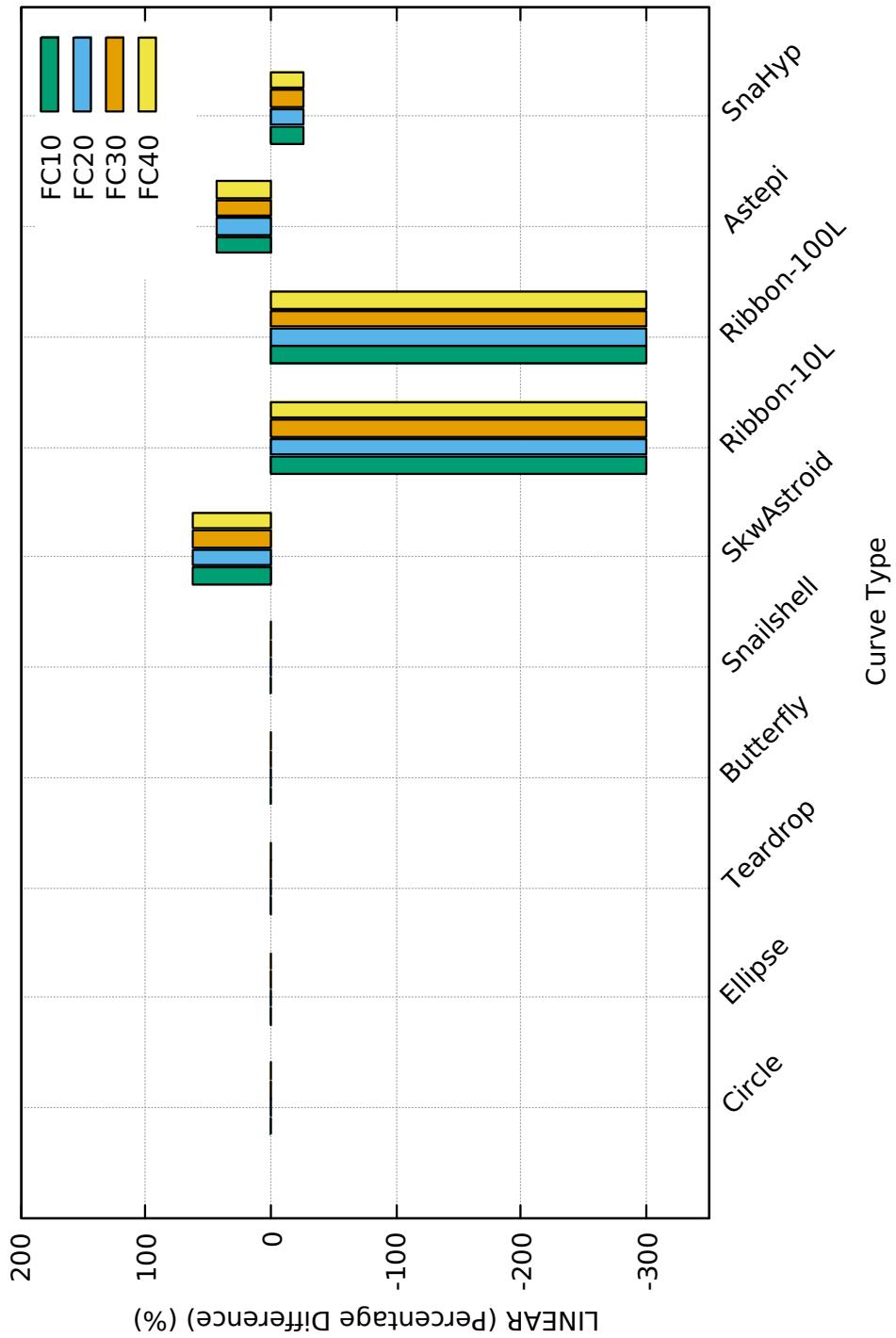


Figure 4.69: Histogram Log10 Overall Percentage Difference between SAL and SCL

Overall Percentage Difference $100 * (\text{SAL-SCL}) / \text{SAL}$ Log 10 Scale for each curve
 at Feedrate Commands FC10, FC20, FC30 and FC40
 with all lambda = 0.18

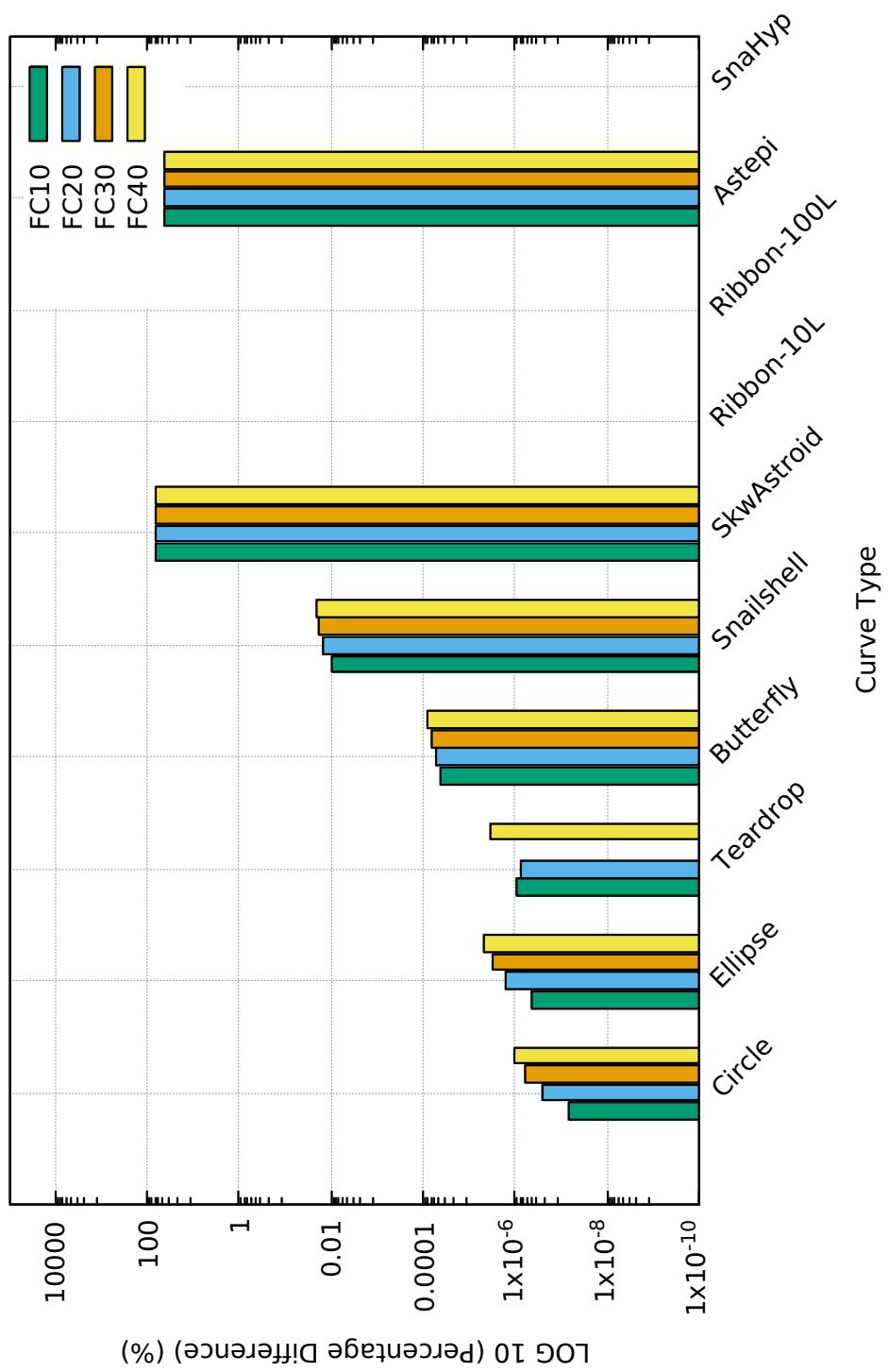


Table 4.25: Overall Percentage Difference between SAL and SCL

OVERALL PCNT DIFF (SAL-SCL)			
Description	PcntDiff (SAL-SCL)	PcntDiff (SAL-SCL)	PcntDiff (SAL-SCL)
Lambda Acceleration Safety Factor	0.18	0.18	0.18
Feedrate Command (mm/s)	FC10	FC20	FC30
Curve Type			
1 Circle	6.652574786747E-08	2.669675295946E-07	6.006449648363E-07
2 Ellipse	4.533921870933E-07	1.654645936542E-06	3.196048378596E-06
3 Teardrop	8.931942058845E-07	7.664163788301E-07	-2.921101261067E-06
4 Butterfly	4.159710526812E-05	5.237880543932E-05	6.611735799000E-05
5 Snailshell	9.396425943836E-03	1.512841599395E-02	1.829675338949E-02
5 Skewed-Astroid	6.163739390653E+01	6.163758077028E+01	6.163776660469E+01
7 Ribbon-10L	-2.999999941577E+02	-2.999999919218E+02	-3.000000205782E+02
8 Ribbon-100L	-2.999999927099E+02	-2.999999893265E+02	-3.000000209086E+02
9 AstEpi	4.410754359858E+01	4.410822100426E+01	4.411017113009E+01
10 SnaHyp	-2.673376648446E+01	-2.673289365118E+01	-2.673239467770E+01

4.8.9 Overall Total Sum-Chord-Error (SCE)

The histogram for the total Sum-Chord-Error (SCE) for all ten(10) curves in this work is provided in Figure [4.70] on the next page. The data table corresponding to the histogram is provided in Table [4.26].

The Overall total sum-chord-error (SCE) on its own is not a good comparison for curve error performance because each curve spans a different curve length. The meaningful measure would be the ratio of total sum-chord-error divided by total sum-chord-length (SCE/SCL). This is akin to the comparison measure of speed in running (meters per second), in which it does not matter how many meters you run how long a time you run. This information on (SCE/SCL) is provided in the section under Chord-Error-Efficiency (CEE) measure in [4.8.10].

It is important to know that $\text{CEE} = \text{SCE}/\text{SCL} = \text{ratio (mm/mm)}$ is unitless. It means the amount of error generated by the algorithm per unit length of chord traversed. This is universally meaningful and applicable to the efficiency of the algorithm.

As expected, generally as Feedrate Command (FC) increases for FC10, FC20, FC30 and FC40, the chord-lengths increases thus increasing the chord-error. That is what the histogram shows. However, two(2) special cases, the Ribbon-10L and Ribbon-100L, seem to be aberrations to this general rule. It has been handled in the Appendices [App.9] and [App.10], respectively.

Figure 4.70: Histogram Overall Sum-Chord-Error SCE

Overall Total Sum-Chord-Error SCE for each curve
at Feedrate Commands FC10, FC20, FC30 and FC40
with all lambda = 0.18

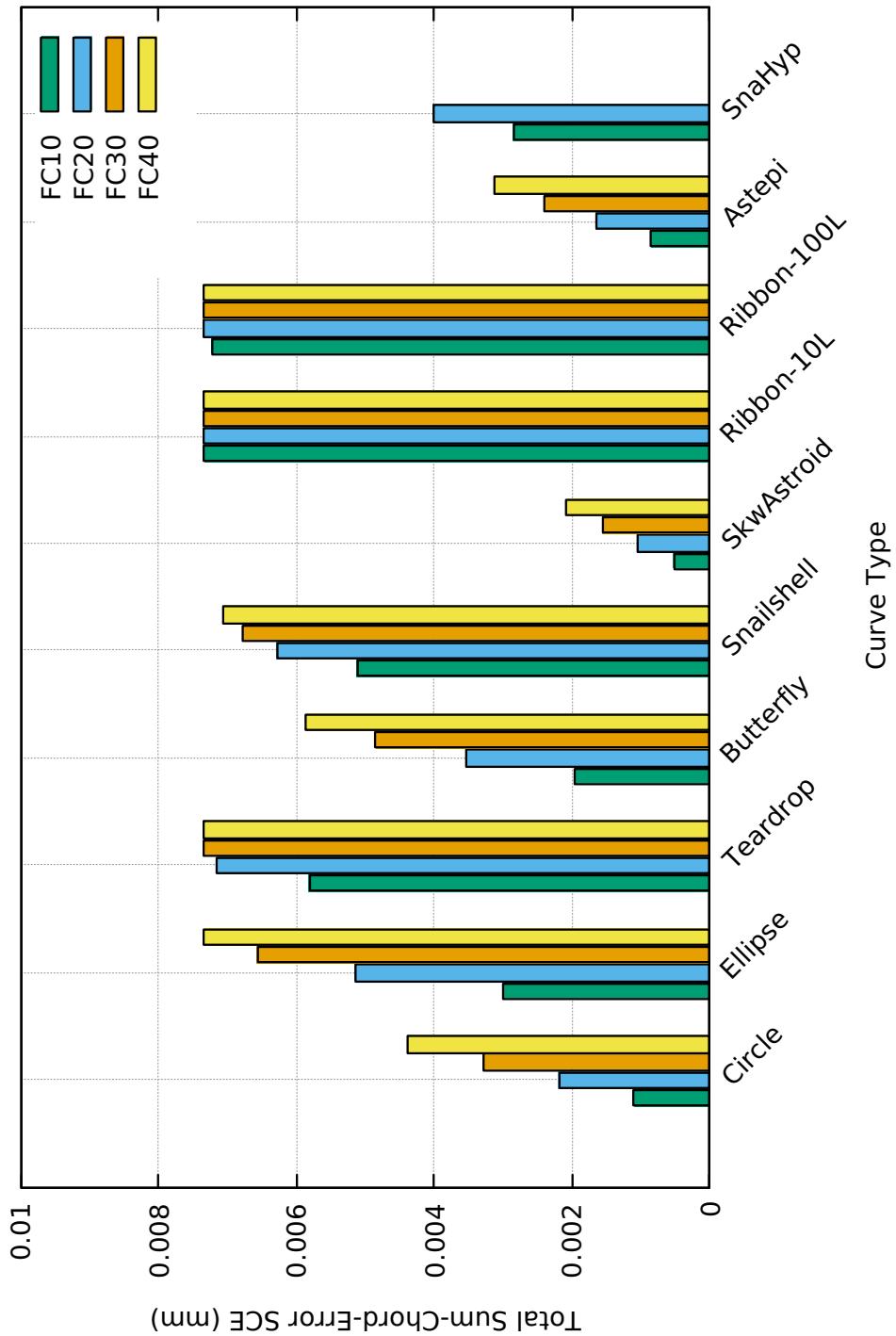


Table 4.26: Overall Sum-Chord-Error Data SCE

OVERALL SUM-CHORD-ERROR (SCE)			
CurveType	Description	Sum-Chord-Error 0.18	Sum-Chord-Error 0.18
1	Circle	1.093913738449E-03	2.187639876000E-03
2	Ellipse	2.990951697698E-03	5.1486611244856E-03
3	Teardrop	5.809737838076E-03	7.140807162860E-03
4	Butterfly	1.938859834664E-03	3.534046223178E-03
5	Snailshell	5.115139395656E-03	6.269762299809E-03
6	Skewed-Astroid	5.163683043374E-04	1.032591177038E-03
7	Ribbon-10L	7.331686925442E-03	7.330650001960E-03
8	Ribbon-100L	7.227413654822E-03	7.334488978808E-03
9	AstEpi	8.403732211685E-04	1.641842648547E-03
10	SnaHyp	2.846873175820E-03	4.002975369043E-03
		0.000000000000E+00	0.000000000000E+00
		Sum-Chord-Error FC10	Sum-Chord-Error FC20
		Sum-Chord-Error FC30	Sum-Chord-Error FC40
		Sum-Chord-Error 0.18	Sum-Chord-Error 0.18
		Sum-Chord-Error 0.18	Sum-Chord-Error 0.18

4.8.10 Overall Chord-Error-Efficiency (CEE)

The histogram for the overall Chord-Error-Efficiency (CEE) for all ten(10) curves in this work is provided in Figure [4.71] on the next page. The data table corresponding to the histogram is provided in Table [4.27].

It is important to know that $\text{CEE} = \text{SCE}/\text{SCL}$ = ratio (mm/mm) is unitless. It essentially means the amount of error generated by the algorithm per unit length of chord traversed. This is universally meaningful and applicable to the efficiency of the algorithm.

The Chord-Error-Efficiency (CEE) is the most important metric for the performance assessment of the algorithm in this thesis.

Except for the Ribbon-10L curve which is tiny (4 mm x 4 mm), many times smaller than the surface area of a fingertip, the other nine(9) curves showed promising values of CEE ranging from the Skewed-Astroid to the Teardrop curves.

Technically, the range for CEE spans from the lowest in Skewed-Astroid to the highest in Teardrop, meaning:

$$(1.158518631091E - 06 \leq \text{CEE} \leq 7.202932786929E - 05).$$

For rounded figures, it shall be considered that

$$(1E - 6 \leq \text{CEE} \leq 7E - 5)$$

is the CEE for the algorithm performance efficiency in this work. That is the take from this section of the thesis.

Figure 4.71: Histogram Overall Chord-Error-Efficiency CEE

Overall Chord-Error Efficiency CEE for each curve
at Feedrate Commands FC10, FC20, FC30 and FC40
with all lamda = 0.18

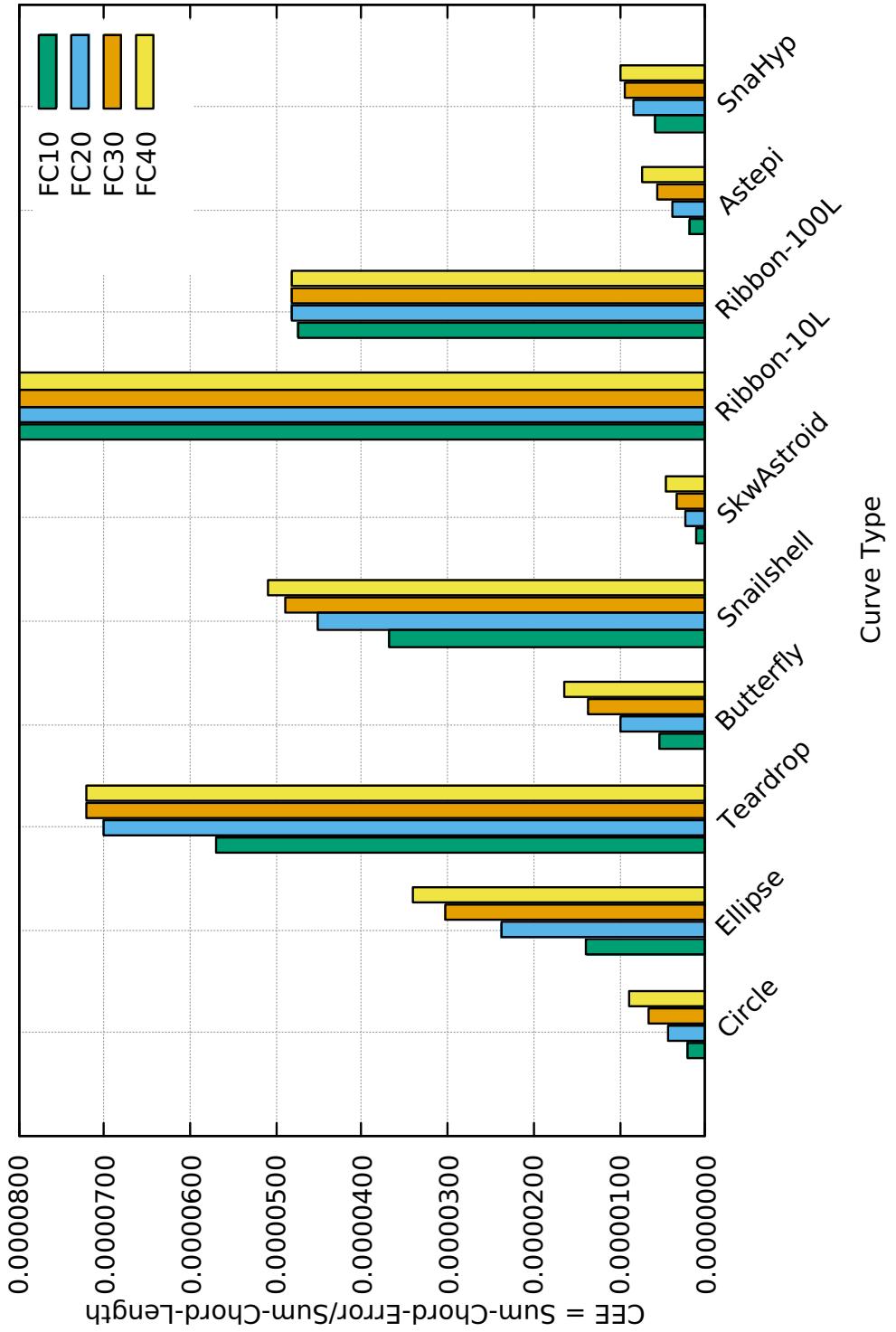


Table 4.27: Overall Performance 2 Chord-Error-Efficiency CEE

Curve Type	CEE PERFORMANCE		sum-chord-error/ sum-chord-length Lamda 0.18	sum-chord-error/ sum-chord-length Lamda 0.18	sum-chord-error/ sum-chord-length Lamda 0.18	sum-chord-error/ sum-chord-length FC40
	FC10	FC20				
1 Circle	2.203789163406E-06	4.407213023407E-06	6.610271383220E-06	8.812956271960E-06		
2 Ellipse	1.386987988559E-05	2.387508767166E-05	3.042915776933E-05	3.399976598039E-05		
3 Teardrop	5.705012452441E-05	7.011661778683E-05	7.202851879506E-05	7.202932786929E-05		
4 Butterfly	5.445092899523E-06	9.925041122094E-06	1.361121273884E-05	1.643334424114E-05		
5 Snailshell	3.691654413053E-05	4.524898511695E-05	4.881992306637E-05	5.085042486059E-05		
6 Skewed-Astroid	1.158518631091E-06	2.316710975042E-06	3.474576811966E-06	4.6322116113725E-06		
7 Ribbon-10L	4.820054776519E-04	4.819406921788E-04	4.818512589475E-04	4.818661991448E-04		
8 Ribbon-100L	4.751833890898E-05	4.822057586226E-05	4.821346349097E-05	4.820475130965E-05		
9 AstEpi	1.971493430212E-06	3.851719659103E-06	5.606883043752E-06	7.299193359018E-06		
10 SnaHyp	5.943528031539E-06	8.356965006276E-06	9.309385400583E-06	9.812009964860E-06		

4.8.11 Overall Total Sum-Arc-Area (SAA)

The histogram for the overall Sum-Arc-Area (SAA) for all ten(10) curves in this work is provided in Figure [4.72] on the next page. The data table corresponding to the histogram is provided in Table [4.28]. To span the entire y-axis for the area, a log base 10 plot is also provided in [4.73]

The Sum-Arc-Area (SAA) is the sum of the areas bounded by the arc segment and the linear chord. The chord-error is the length of the perpendicular line from the center of the chord that intersects the arc segment. Therefore, as the chord length increases, the chord-error increases, and so the arc-area increases too.

Even though the arc-area is directly proportional to the chord-error, the overall total Sum-Arc-Area (SAA) on its own is not a good comparison for curve arc-area performance because each curve spans a different curve length.

The meaningful measure would be the ratio of total Sum-Arc-Area (SAA) divided by total Sum-Chord-Length (SCL). The SCL is used as a base because it is an exact and reliable measure. Thus, Arc-Area-Efficiency AAE = (SAA/SCL). This is akin to the comparison measure of how much "bounded area" the algorithm generates per unit chord-length traversed. It does not matter how many meters of chord-length traversed for any curve because it is a measure per unit length.

The information on (SAA/SCL) is provided in the next section under Arc-Error-Efficiency (AEE) in [4.8.12].

Figure 4.72: Histogram Overall Sum-Arc-Area SAA

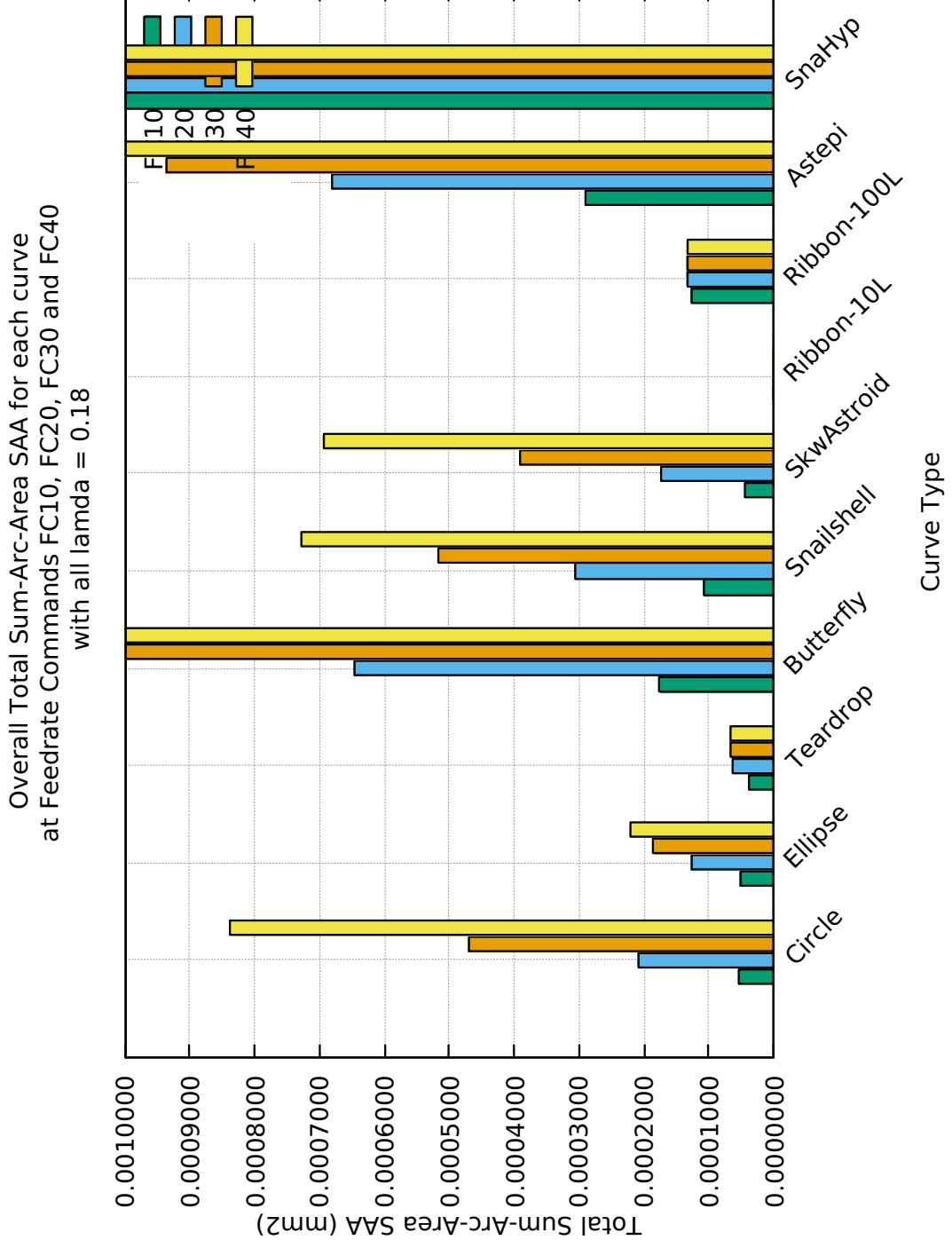


Figure 4.73: Histogram Log 10 Overall Sum-Arc-Area SAA

Overall Total Sum-Arc-Area SAA on Log 10 for each curve
at Feedrate Commands FC10, FC20, FC30 and FC40
with all lambda = 0.18

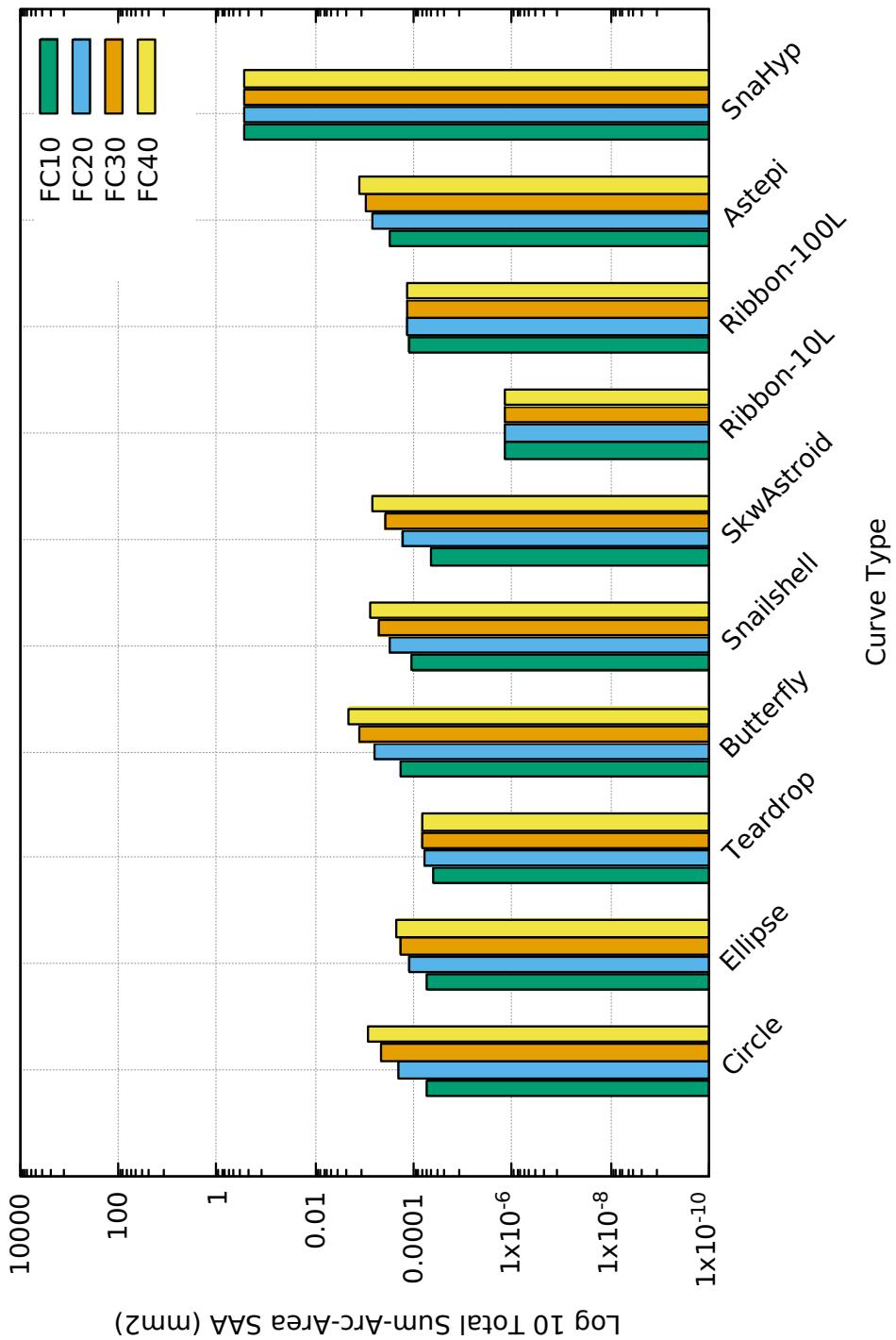


Table 4.28: Overall Sum-Arc-Area SAA

Curve Type	SAA PERFORMANCE			Sum-Arc-Area (mm2) Lamda 0.18 FC10	Sum-Arc-Area (mm2) Lamda 0.18 FC20	Sum-Arc-Area (mm2) Lamda 0.18 FC30	Sum-Arc-Area (mm2) Lamda 0.18 FC40	Sum-Arc-Area (mm2) Lamda 0.18					
1	Circle	5.235292684461E-05	2.093803820914E-04	4.710353817982E-04	8.373046676549E-04								
2	Ellipse	5.185017589366E-05	1.275076043292E-04	1.849039783361E-04	2.200267776119E-04								
3	Teardrop	3.822127588087E-05	6.182290957317E-05	6.781012634326E-05	6.777759239889E-05								
4	Butterfly	1.758667624644E-04	6.462621773264E-04	1.298932073590E-03	2.003665777961E-03								
5	Snailshell	1.070830851582E-04	3.068808285119E-04	5.164360050036E-04	7.283849492346E-04								
5	Skewed-Astroid	4.343191363991E-05	1.736583480748E-04	3.905753943512E-04	6.940792041172E-04								
7	Ribbon-10L	1.338515095667E-06	1.338121185915E-06	1.338217831316E-06	1.337920946871E-06								
8	Ribbon-100L	1.265139232664E-04	1.339597112819E-04	1.339309239988E-04	1.339343540720E-04								
9	AstEpi	2.898515736808E-04	6.811935608835E-04	9.367445601128E-04	1.255001527807E-03								
10	SnaHyp	2.706870610425E-01	2.76007501922E-01	2.777614243305E-01	2.754368695310E-01								

4.8.12 Overall Arc-Area-Efficiency (AAE)

The histogram for the overall Arc-Area-Efficiency (AAE) for all ten(10) curves in this work is provided in Log base 10 scale in Figure [4.74] on the next page. The data table corresponding to the histogram is provided in Table [4.29].

The Arc-Area-Efficiency (AAE) is defined as the ratio (SAA/SCL). This is the comparison measure of how much "bounded area" the algorithm generates per unit chord-length traversed. It does not matter how many meters of chord-length traversed for any curve because it is a measure per unit length.

The Arc-Area-Efficiency (AAE) is the second most important metric, after CEE, for the performance assessment of the algorithm in this thesis. This is due to the approximate calculation for the arc-area, especially for curves other than simple curves like the Circle and Ellipse curves.

Except for the Ribbon-10L curve which is tiny (4 mm x 4 mm), many times smaller than the surface area of a fingertip, the other nine(9) curves showed promising values of AAE ranging from the Skewed-Astroid to the Teardrop curves.

Technically, ignoring the Ribbon-10L curve, the range for AAE spans from the lowest in Skewed-Astroid to the highest in Teardrop, meaning:

$$(1.158518631091E - 06 \leq AAE \leq 7.202932786929E - 05)$$

The range above is exactly the same for CEE where:

$$(1.158518631091E - 06 \leq CEE \leq 7.202932786929E - 05).$$

For rounded figures, it shall be considered that

$$(1E - 6 \leq AEE \leq 7E - 5) \text{ or}$$

Figure 4.74: Histogram Log 10 Overall Arc-Area-Efficiency AAE

Overall Arc-Area-Efficiency AAE for each curve
at Feedrate Commands FC10, FC20, FC30 and FC40
with all lambda = 0.18

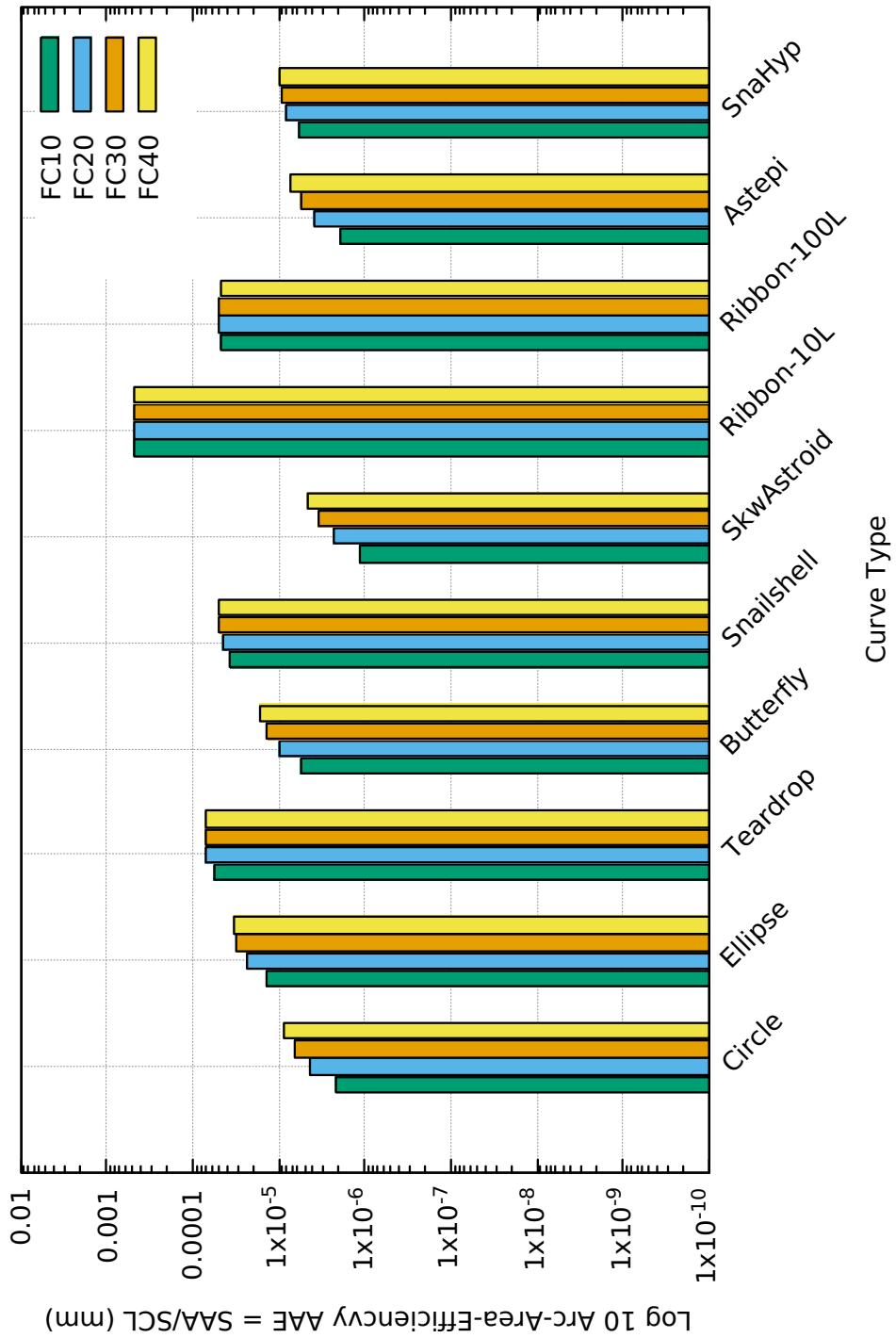


Table 4.29: Overall Arc-Area-Efficiency AAE

OVERALL ARC-AREA-EFF (AAE)		Description	Arc-Area-Eff	Arc-Area-Eff	Arc-Area-Eff
	Lambda Safety Factor	0.18	0.18	0.18	0.18
	Feedrate Command (mm/s)	FC10	FC20	FC30	FC40
Curve Type					
1	Circle	2.203789163406E-06	4.407213023407E-06	6.610271383220E-06	8.812956271960E-06
2	Ellipse	1.386987988559E-05	2.387508767166E-05	3.042915776933E-05	3.399976598039E-05
3	Teardrop	5.705012452441E-05	7.011661778683E-05	7.202851879506E-05	7.202932786929E-05
4	Butterfly	5.445092899523E-06	9.925041122094E-06	1.361121273884E-05	1.643334424114E-05
5	Snailshell	3.691654413053E-05	4.524898511695E-05	4.881992306637E-05	5.085042486059E-05
5	Skewed-Astroid	1.158518631091E-06	2.316710975042E-06	3.474576811966E-06	4.632116113725E-06
7	Ribbon-10L	4.820054776519E-04	4.819406921788E-04	4.818512589475E-04	4.818661991448E-04
8	Ribbon-100L	4.751833890898E-05	4.822057586226E-05	4.821346349097E-05	4.820475130965E-05
9	AstEpi	1.971493430212E-06	3.851719659103E-06	5.606883043752E-06	7.299193359018E-06
10	SnaHyp	5.943528031539E-06	8.356965006276E-06	9.309385400583E-06	9.812009964860E-06

4.9 SUMMARY OF RESULTS CHAPTER

1. The objective of this thesis was illustrated in a real live run of the G-code generated by the parametric curve interpolation algorithm in section [4.3.3]. This section describes the final outputs of the work in this thesis. The main objective of the algorithm execution is to ensure that the resulting running feedrate (speed motion of the cutting tool) is smooth and continuous, does not exceed the feedrate limit, at the same time maintains chord-error below tolerance, while traversing the full curve path.
2. The rationale for the Teardrop parametric curve selected as the illustrative curve is described in section [4.4]. The results of algorithm execution on the rest of the nine(9) parametric curves are provided in the respective appendices.
3. The descriptive comparisons of the characteristics of the other nine(9) parametric curves against the Teardrop curve are provided in section [4.4.2]. The additional challenges these nine(9) curves impose on the algorithm were also discussed.
4. The absolute constraint on chord-error to be below error-tolerance ($1E-6$) was discussed in section [4.5.7], and the absolute constraint on the current or running feedrate to be just below the feedrate limit was discussed in section [4.5.10]. These two(2) constraints are the foundations of this thesis.
5. In order to execute the interpolation algorithm three(3) user inputs are required, namely: curve type, feedrate command FC, and lamda safety acceleration factor. Ten(10) different parametric curves can be selected. The Feedrate Command FC, for execution runs are specified for FC10, FC20, FC30 and FC40 based on the rationale discussed in section [4.8.1].
6. The acceleration safety factor lamda determines not just the safety limits of the algorithm execution with regards to the physical CNC machine, but also determines the onset of acceleration jitters, the occurrence of which is absolutely not acceptable. In section [4.3.4], the value of lamda was found to be lamda = 0.18, which is

considered the threshold for acceleration safety factor. Any lamda value above 0.18 will cause tangential acceleration jitters, that translates to jerky running feedrates. The smoothness requirement on running feedrate and the absolute confinement of tangential acceleration discussed in section [4.5.19], are some of the main objectives for the interpolation algorithm in this thesis.

7. The presentation of Teardrop specific data under nominal run at FC20 and Lamda 0.18, covers the radius of curvature, first and second-order terms of Taylor's approximation, chord-error absolute constraint, the four components of feedrate limit, current feedrate absolute constraint, histogram distribution of interpolated points, rising and falling feedrate profiles, feedrate profile for the main region, color-coded running feedrates, algorithm performance on the curve and, tangential acceleration profiles.
8. For comparisons, the summary of algorithm performance data in landscape mode for all ten(10) curves are provided covering the Circle, Ellipse, Teardrop, Butterfly, Snailshell, Skewed-Astroid, Ribbon-10L, Ribbon-100L, AstEpi and SnaHyp curves in section [4.6].
9. To assess the performance of the interpolation algorithm, four(4) ratio metrics were created comprising namely: Total Interpolation Points per unit chord-length traversed denoted by (TIP/SCL), the total Sum-Chord-Error (SCE) per unit chord-length traversed denoted by (SCE/SCL), and the total Sum-Arc-Area (SAA) per unit chord-length traversed denoted by (SAA/SCL). The total Sum-Chord-Length(SCL) was used as the base metric because the calculation for total Sum-Chord-Length (SCL) is exact (length of line between two (x,y) points), while the total Sum-Arc-Length (SAL) is just a first-order arc-length approximation based on arc segments.
10. It was found that the Sum-Arc-Length (SAL) calculation is good and acceptable for five(5) curves comprising the Circle, Ellipse, Teardrop, Butterfly and Snailshell curves, and is bad thus unacceptable for the other five(5) curves comprising the Skewed-Astroid, Ribbon-10L, Ribbon-100L, AstEpi and SnaHyp curves. For this

reason, instead of the total Sum-Arc-Length (SAL), the total Sum-Chord-Length (SCL) was used as the base metric.

11. The Chord-Error-Efficiency (CEE) is defined as (SCE/SCL) and interpreted as the amount of chord-error generated by the algorithm per unit of chord-length traveled. The Arc-Area-Efficiency (AAE) is defined as (SAA/SCL) and interpreted as the amount of "arc-bounded area" generated by the algorithm per unit of chord-length traveled. It was found as expected, that the CEE is directly proportional to AAE, and in some cases exactly equal to each other.
12. In section [4.8.10], the range for Chord-Error-Efficiency (CEE), was found to be $(1.158518631091E - 06 \leq CEE \leq 7.202932786929E - 05)$.
13. In section [4.8.12], the range for Arc-Area-Efficiency (AAE), was found to be $(1.158518631091E - 06 \leq AAE \leq 7.202932786929E - 05)$.
14. The measurement of execution time of the algorithm on the computer (in seconds) is not a true metric but just an indication of the amount of internal calculations (iterative and recursive optimizations) occurring inside of the algorithm.

Chapter 5

CONCLUSION

5.1 Conclusions of the research

The realtime interpolation algorithm developed in this work, when executed against all of the selected curves exclusively and simultaneously satisfy both the design constraints on the feedrate and its chord-error tolerance. The summary of work achievements are as follows.

1. The resulting feedrate profiles throughout the entire path are continuous and smooth.
2. The maximum feedrate at every interpolated point remains below and close to the calculated feedrate limit.
3. The current running feedrate does not exceed the user specified command feedrate at every point during the traversal along the entire parametric curve.
4. The chord-error at every interpolated arc segment remains below the chord-error tolerance throughout the curve traversal.
5. Without fail, a single and robust interpolation algorithm was developed and is capable of handling the complexities of a wide range of 2D parametric curves comprising different sizes and shapes.
6. The algorithm can be implemented both in a realtime, online mode by driving the CNC machine directly, or in an offline mode by using a stored RS274/NGC G-code

standard file. This work satisfies both interpretations of realtime: the lay person interpretation and the technical definition of realtime.

7. The performance accuracy of the algorithm for the chord-error per unit length traversed and error-tolerance set at $(10)^{-6}$ mm, as expected, varies for different curves.
8. The average performance of chord-error to length ratio is in the range of $(10)^{-6}$ to $(10)^{-5}$ for the algorithm. The results are provided in [Table 5.1].

5.2 Performance accuracy of the algorithm

The curve-error per unit length traversed is the sum total of all chord-errors divided by the sum total of all chord lengths for each parametric curve. This ratio is a performance accuracy measure of the interpolation algorithm developed in this work.

Table 5.1: Chord-error per unit length curve traversed (ratio) CEE

Curve Type	Total Chord Length(mm)	CEE	CEE	CEE
		Ratio Min	Ratio Avg	Ratio Max
1 Teardrop curve	101.83	$5.70(10)^{-5}$	$6.45(10)^{-5}$	$7.20(10)^{-5}$
2 Butterfly curve	356.07	$5.45(10)^{-6}$	$1.09(10)^{-5}$	$1.64(10)^{-5}$
3 Ellipse curve	215.64	$1.38(10)^{-5}$	$2.39(10)^{-5}$	$3.40(10)^{-5}$
4 SkewedAstroid curve	445.71	$1.16(10)^{-6}$	$2.89(10)^{-5}$	$4.62(10)^{-6}$
5 Circle curve	496.39	$2.20(10)^{-6}$	$4.75(10)^{-6}$	$7.30(10)^{-6}$
6 AstEpi curve	426.26	$1.97(10)^{-6}$	$5.39(10)^{-6}$	$8.81(10)^{-6}$
7 Snailshell curve	138.56	$3.69(10)^{-5}$	$4.38(10)^{-5}$	$5.08(10)^{-5}$
8 SnaHyp curve	478.99	$5.94(10)^{-6}$	$7.62(10)^{-6}$	$9.31(10)^{-6}$
9 Ribbon10L curve	15.21	$4.81(10)^{-4}$	$4.82(10)^{-4}$	$4.82(10)^{-4}$
10 Ribbon100L curve	152.13	$4.75(10)^{-5}$	$4.79(10)^{-5}$	$4.82(10)^{-5}$

The ratio in the table above is termed Chord-Error-Efficiency (CEE) and interpreted as the amount of error generated per unit chord-length of curve travel. This CEE ratio is independent of the total length of the curve.

Ignoring the Ribbon10L curve for being too small (4 mm x 4 mm), the average performance of chord-error to length ratio is in the range of $(10)^{-6}$ to $(10)^{-5}$ for the algorithm developed in this work.

5.3 Knowledge contributions

1. A parametric curve interpolation method that successfully constrain chord-error and feedrate simultaneously with confinement of tangential acceleration.
2. The importance of having a fully functionalized and modularized implementation code for interpolation that follows good software engineering practice.

5.4 Recommendations for future work

It is highly recommended that future interpolation work involves NURBS curves. The availability of libraries in various programming languages to manipulate NURBS, like C/C++, Python, Julia and Octave, opens up possibilities for commercial applications of NURBS in CNC machining. It is recommended that a study be conducted to compare the simplicity, accuracy and intensiveness of computations using NURBS libraries of the different programming languages.

REFERENCES

- Abdelgader, M. S. A., & Yusoff, W. R. (2014). *Driving Computer Numerical Controlled (CNC) Machine Using PC, Heber x10i, and Raspberry pi 2* (Tech. Rep.). Faculty of Computing and Informatics: Final Year Project, Multimedia University, Cyberjaya.
- Ahmad, A. H., & Yusoff, W. R. (2017). *Using Raspberry Pi 3 Model B to drive a CNC system in Real Time* (Tech. Rep.). Faculty of Computing and Informatics: Final Year Project, Multimedia University, Cyberjaya.
- Ariffin, H., & Yusoff, W. R. (2014). *Using Arduino Due to drive a CNC system* (Tech. Rep.). Faculty of Computing and Informatics: Final Year Project, Multimedia University, Cyberjaya.
- Bhattacharjee, B., Azeem, A., Ali, S. M., & Paul, S. K. (2012). Development of a CNC interpolation scheme for CNC controller based on Runge-Kutta method. *International Journal of Computer Aided Engineering and Technology Vol 4 No 5*, 445-464.
- Bingol, O. R., & Krishnamurty, A. (2018). NURBS-Python: An open-source object-oriented NURBS modeling framework in Python. *SoftwareX 9 - Elsevier*, 85-94. <https://doi.org/10.1016/j.softx.2018.12.005>
- B Shengzhou, X. S. H. Z., C Han, & Jiang, Y. (2020). Teardrop hovering formation for elliptical orbit considering J2 perturbation. *Aerospace Science and Technology, Vol. 106, 106098*.
- Chen, J., & Guo, R. (2023). Stack and Heap Memory, Notes of data structures and C++ concepts. *Course Notes, Last retrieved: 17 September 2023*. Retrieved from <https://courses.engr.illinois.edu/cs225/fa2022/resources/stack-heap/>
- Chen, M., & Sun, Y. (2019). Contour error-bounded parametric interpolator with minimum feedrate fluctuation for five-axis CNC machine tools. *International Journal of Advanced Manufacturing Technology 103*, 567-584. <https://doi.org/10.1007/s00170-019-03586-5>
- Cheng, M.-Y., Tsai, M. C., & Kuo, J. C. (2002). Real-time NURBS command generators for CNC Servo Controllers. *International Journal of Machine Tools and Manufacture 42(7)*, 801-813. [https://doi.org/10.1016/S0890-6955\(02\)00015-9](https://doi.org/10.1016/S0890-6955(02)00015-9)
- CollabCNC, A. A. (2023). How do you integrate NURBS with other CNC programming methods and techniques. *LinkedIn.com, Last updated on Jul 27, 2023*. Retrieved from <https://www.linkedin.com/advice/3/how-do-you>

-integrate-nurbs-other-cnc-programming

- CollabCNC, B. A. (2023). How do you evaluate the accuracy and quality of NURBS machining results. *Linkedi.com, Last updated on Jul 27, 2023.* Retrieved from <https://www.linkedin.com/advice/3/how-do-you-evaluate-accuracy-quality-nurbs-machining>
- Farouki, R., & Tsai, Y. (2001). Exact Taylor series coefficients for variable-feedrate CNC curve interpolators. *Computer-Aided Design* 33, 155-165. [https://doi.org/10.1016/S0010-4485\(00\)00085-3](https://doi.org/10.1016/S0010-4485(00)00085-3)
- Farouki, R. T. (2021). Accurate Real-time CNC Curve Interpolators Based Upon Richardson Extrapolation. *Computer-Aided Design* 135. <https://doi.org/10.1016/j.cad.2021.103005>
- Fu, H., Li, C., & Fu, Y. (2016). A parallel CNC system architecture based on Symmetric Multi-processor. *Sixth International Conference on Instrumentation and Measurement, Computer, Communication and Control*, 634-637. <https://doi.org/10.1109/IMCCC.2016.74>
- Hu, Y., Jiang, X., Huo, G., Su, C., Li, H., & and, Z. Z. (2023). A novel method for calculating interpolation points of NURBS curves based on chord length-parameter ratio. *Research Square*, 155-165. <https://doi.org/10.21203/rs.3.rs-3155656/v>
- Jia, Z. Y., Ma, J. W., Song, D. N., Wang, F. J., & Liu, W. (2018). A review of contouring-error reduction method in multi-axis CNC machining. *International Journal of Machine Tools and Manufacture* 125, 34-54. <https://doi.org/10.1016/j.ijmachtools.2017.10.008>
- Jin, Y., He, Y., Fu, J.-Z., Lin, Z.-W., & Gan, W.-F. (2014). A fine-interpolation-based parametric interpolation method with a novel real-time look-ahead algorithm. *Computer-Aided Design* 55, 37-48. <https://doi.org/10.1016/j.cad.2014.05.002>
- Kalimbetov, A., & Yusoff, W. R. (2012). *C/C++ and Python for Linux Realtime Parallel Port Software driver* (Tech. Rep.). Faculty of Computing and Informatics: Final Year Project, Multimedia University, Cyberjaya.
- Koren, Y. (1976). Interpolator for a CNC system. *IEEE Transactions on Computers*, Vol. C 25, 32-37.
- Koren, Y., Lo, C. C., & Shpitalni. (1993). CNA INTERPOLATORS: ALGORITHMS AND ANALYSIS. *Manufacturing Science and Engineering PED* Vol. 64, 83-92.
- Lee, C.-Y., Kim, S. H., Ha, T. I., Min, J., Hwang, S.-H., & Min, B.-K. (2018). CNC Algorithms for Precision Machining: State of the Art Review. *Journal of the Korean Society for Precision Engineering*, vol. 35, 279-291.
- Nam, S.-H., & Yang, M.-Y. (2004). A study on a generalized parametric interpolator with real-time jerk-limited acceleration. *Computer-Aided Design* 36, 27-36.
- Ni, H., Zhang, C., Chen, C., Hu, T., & Liu, Y. (2019). A parametric interpolation method based on prediction and iterative compensation. *International Journal of Advanced*

Robotic Systems, 1-10. <https://doi.org/10.1177/1729881419828188>

Piegl, L., & Tiller, W. (1997). The NURBS Book, 2nd edition. In (p. 1-43). Springer-Verlag.

Quang, N. H., & Long, B. T. (2017). A Method of Real-Time NURBS Interpolation with Confined Chord Error for CNC Systems. *Vietnam Journal of Science and Technology* 55 (5), 650-657.

Ramesh, R., Mannan, M. A., & Poo, A. N. (2005). Tracking and contour error control in CNC servo systems. *International Journal of Machine Tools and Manufacture* 45, 301-326. <https://doi.org/10.1016/j.ijmachtools.2004.08.008>

Rob, W. (2022). *Smooth Trajectory Generation for 5-Axis CNC Machine Tools* (Unpublished doctoral dissertation). Ph. D Thesis of University of Sheffield, U.K.

Rogers, D. F. (2001). An Introduction to NURBS with Historical Perspective. In (p. 1-11). Academic Press.

Selvarajah, R., & Yusoff, W. R. (2015). *Using the Beagle Board xM SBC to drive a CNC system* (Tech. Rep.). Faculty of Computing and Informatics: Final Year Project, Multimedia University, Cyberjaya.

Shpitalni, H., Koren, Y., & Lo, C. C. (1994). Realtime curve interpolators. *Computer Aided Design Vol 26 No 11*, 832-838.

S.H. Suh, D. C., S.K. Kang, & Stroud, I. (2008). Theory and design of CNC systems. In (p. 1-454). Springer Sciences and Business Media.

S. Sun, C. W., D. Yu, & Xie, C. (2018). A smooth tool path generation and real-time interpolation algorithm based on B-spline curves. *Advances in Mechanical Engineering.vol.10(1)*, 650-657. <https://doi.org/10.1177/1687814017750281>

Sun, Y., Wang, J., & Guo, D. (2006). Guide curve based interpolation scheme of parametric curves for precision CNC machining. *International Journal of Machine Tools and Manufacture* 46, 235-242. <https://doi.org/10.1016/j.ijmachtools.2005.05.024>

Teh, C., & Yusoff, W. R. (2016). *Using the Nexys-3 Spartan-6 FPGA board to develop a closed-loop feedback CNC system* (Tech. Rep.). Faculty of Computing and Informatics: Final Year Project, Multimedia University, Cyberjaya.

Thomas R. Kramer, F. M. P., & Messina, E. (2000). The NIST-RS274NGC-Intrepreter- Version 3. In (p. 1-121). National Institute of Standards and Technology.

TitanCNC, A. (2021). Incredible 9 Axis Machining on DN solutions puma SMX3100ST [video]. *You Tube Channel, dated 13 April, 2021*. Retrieved from <https://www.youtube.com/watch?v=HH9FdfMcpI0>

Wikipedia, A. A. (2023). Stack and Heap Memory, Notes of data structures and C++ concepts. *Wikipedia, Last retrieved: 17 September 2023*. Retrieved from https://en.wikipedia.org/wiki/Stack-based_memory_allocation

Wikipedia, B. A. (2023). Machine epsilon (computer science). *Wikipedia, Last edited:*

04 September 2023. Retrieved from https://en.wikipedia.org/wiki/Machine_epsilon

Wikipedia, C. A. (2023). Side effect (computer science). *Wikipedia*, Last edited: 03 July 2023. Retrieved from [https://en.wikipedia.org/wiki/Side_effect_\(computer_science\)](https://en.wikipedia.org/wiki/Side_effect_(computer_science))

Yeh, S.-S. (2019). Feed Rate Determination Method for Tool Path Iterpolation Considering Piecewise-Continued Machining Segments with Cornering Errors and Kinematic Constraints. *International Journal of Mechanical Engineering and Robotics Research* Vol. 8, No. 3.

Yeh, S. S., & Hsu, P. L. (1999). The speed-controlled interpolator for machining parametric curves. *Computer-Aided Design* 31, 349-357.

Yeh, S.-S., & Hsu, P.-L. (2002). Adaptive-feedrate interpolation for parametric curves with a confined chord error. *Computer-Aided Design* 34, 229-237.

Yu B-F, C. J.-S., & Yu. (2020). Development of an Analyzing and Tuning Methodology for the CNC Parameters Based on Machining Performance. *Applied Sciences*. Vol. 10(8):2702. <https://doi.org/10.3390/app10082702>

Zhong, W., Luo, X., Chang, W., Ding, F., & Cai, Y. (2018). A real-time interpolator for parametric curves. *International Journal of Machine Tools and Manufacture* 125, 133-145. <https://doi.org/10.1016/j.ijmachtools.2017.11.010>

APPENDICES

.1 APPENDIX CHAPTER 3

.1.1 About Side Effects in Computations

In computer science, an operation, function or expression is said to have a side effect if it modifies some state variable value(s) outside its local environment, which is to say if it has any observable effect other than its primary effect of returning a value to the invoker of the operation. Source: [Side effect in computer science].

It is extremely important that computing operations do not cause side effects unintentionally. The programmer must be aware of side effects if imperative programming paradigm is implemented, where the application state flows from one function to the next. The unintended changes in state information are basically side effects. If the change is intended (deliberate) then it is not a side effect.

For example, a global variable "count_total_lines" is intended to be incremented every time a specific local function executes, then it is not a side effect. When a second local function executes, it is also intended that the global variable "count_total_lines" be incremented. This is not a problem.

The side effect problems may exist if the program implements many global variables that are inter-related to each other in computations. Since there are many variables involved, it will be difficult to track which variable values should change, and which should not change. This is where side effects may occur. For example, if by mistake successive executions of different functions do not happen in the order as designed, then side effects will definitely occur. One example in computer science that creates side effect is a race condition. It was said that side effects are not only limited to state manipulation, in fact, interacting with the I/O, database, log system, APIs and anything else that can be controlled, has a side effect.

It is known that the functional programming paradigm aims to minimize or eliminate side effects. The lack of side effects in functional programming makes it easier to do formal verification of a program. Formal verification means verifying that the function executes correctly, and the execution also produces the correct results.

In general programming, there are many ways to reduce and minimize the occurrence of side effects. Proper naming of variables that separates global from local variables is helpful. Having the program flow design to be fully structured definitely helps. Program flows that are unstructured or haphazard invites disaster, especially the occurrence of side effects.

In practice, most applications will combined the declarative and imperative programming paradigms. There is a fine balancing act between the declarative (what do to) and imperative (how to do) paradigms, with more a shift in the community towards declarative programming.

This appendix is not meant to provide an exhaustive analysis on side-effects, a good read on it is provided at the URL link [What are side effects and what you can do about them (2020).].

.1.2 Realtime Computing Environment

The official title of this thesis is "*Realtime interpolation of parametric curves with chord-error and feedrate constraints.*" The word "realtime" is the first word in the title and it deserves to be discussed.

To a lay person, realtime means happening now, happening currently or immediately, or as it is happening, or occurring at the current instance or not happening in past time, or not occurring in delayed time. These interpretations are all correct. However, the interpretation of realtime in computing is different. This work satisfies both the layman and technical computing interpretations of realtime.

In layman's interpretation for the algorithm in this work, realtime means that as soon as the computation of the next interpolated point is completed, the interpolation algorithm will immediately send signals to the CNC machine to make that specific move. The cycle repeats for the next interpolated point until it reaches the end of the curve. This is the realtime according to the layman's interpretation.

Technically, in computing, real time is a guaranteed level of computer responsiveness within a specified time constraint, usually milliseconds or microseconds, between an event and its response deadline. This time delay between an event and its response is termed as time latency or just latency.

Real-time or real time describes various operations in computing or other processes that must guarantee response times within a specified time (deadline), usually a relatively short time. A real-time process is generally one that happens in defined time steps of maximum duration and fast enough to affect the environment in which it occurs, such as inputs to a computing system. Realtime does not mean fast or happening now. Realtime means it must be done before its deadline.

The generally accepted technical definition of realtime is provided at URL [Real-time Computing]. Real-time computing (RTC) is the computer science term for hardware and software systems subject to a "real-time constraint", for example from event to system response. Real-time programs must guarantee response within specified time constraints, often referred to as "deadlines" or "maximum latency".

The interpolation algorithm developed in this work runs on a Real-Time Operating System (RTOS) and so makes the word "realtime" in its title accurate. The realtime requirements for timing latency of the algorithm's execution must be satisfied.

The LinuxCNC-Axis software that drives the G-Code signals generated by the algorithm runs on a realtime operating system (RTOS). If it is a non-RTOS, LinuxCNC-Axis software can only simulate running G-codes but not actually driving the electrical signals to the CNC machine. This is the first part of realtime in this thesis.

The next question is about the timing latency of computers in this work that will drive the G-Code signals. The maximum latency or deadline in this work is the time between the event of sending the signal to the time the CNC motors respond with the appropriate move. The signal travel time along the electrical wires is not a problem, since it is about 80 percent of the speed of light (almost instantaneous). The realtime requirement here is about the timing capability of the computer in sending repeated signals (worst case duration between a signal and its next signal). Sending a signal is an event, so the time between successive signals is a measure of latency. This worst case latency must be guaranteed for the computer.

Are the computers in this work capable of guaranteeing worst case latencies or deadlines to be considered realtime? The answer is definitely yes. It is confirmed by actual measurements, in which the results and discussions are provided in this document in section Cyclictest for Computer Latency Measurements [.1.3].

.1.3 Cyclictest for computer latency measurements

In computing community, the cyclictest application program is a standardized shared code test that is used universally to measure computer hardware latencies. The results are captured in a tabular data format and plotted on a histogram. Essentially, the cyclictest code measures latency of the combined effect of the hardware, the firmware, and the operating system.

Cyclictest accurately and repeatedly measures the difference between a thread's intended wake-up time and the time at which it actually wakes up in order to provide statistics about the system's latencies. Cyclictest is most commonly used for benchmarking realtime (RT) systems. It is one of the most frequently used tools for evaluating the relative performance of real-time systems. Source: [Linux Foundation Cyclictest]

The results of cyclictest measurements in microseconds (us) for 10 million cycles (30 minutes run) in this work are as follows:

1. PASSED as RTOS. Histogram [1] shows a maximum latency value 42 (us) for HP-Laptop-01, on OS Debian 10 Kernel: 4.19.0-25-rt-amd64.
2. FAILED as RTOS. Histogram [2] shows a maximum latency value 431 (us) for HP-Laptop-01, on OS Ubuntu 20.04 LTS Kernel: 5.15.0-86-lowlatency.
3. PASSED as RTOS. Histogram [3] shows a maximum latency value 46 (us) for HP-Laptop-02, on OS Debian 10 Kernel: 4.19.0-25-rt-amd64.
4. FAILED as RTOS. Histogram [4] shows a maximum latency value 872 (us) for HP-Laptop-02, on OS Ubuntu 20.04 LTS Kernel: 5.15.0-86-lowlatency.

In the four(4) CPU Latency Performance Summary data tables at [1], [2], [3], and [4], we can see that the average latencies for each of the 8-CPU cores as 2 (us) for RTOS and 8 (us) for non-RTOS. It is not the average latency that determines the RTOS status but the value of the worst case or maximum latency. The lower value for latency the better.

Figure 1: Histogram-Latency-HP-Laptop-01-Debian10-10million-cycles

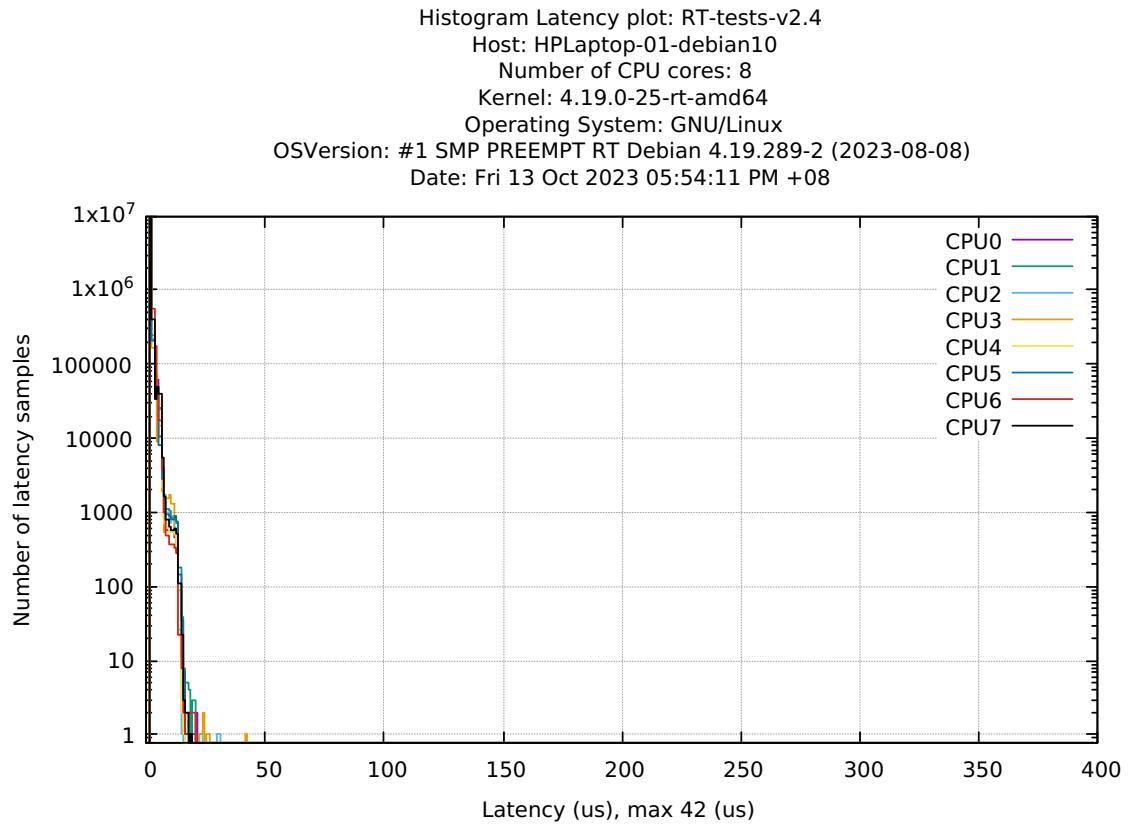


Figure 2: Histogram-Latency-HP-Laptop-01-Ubuntu20-10million-cycles

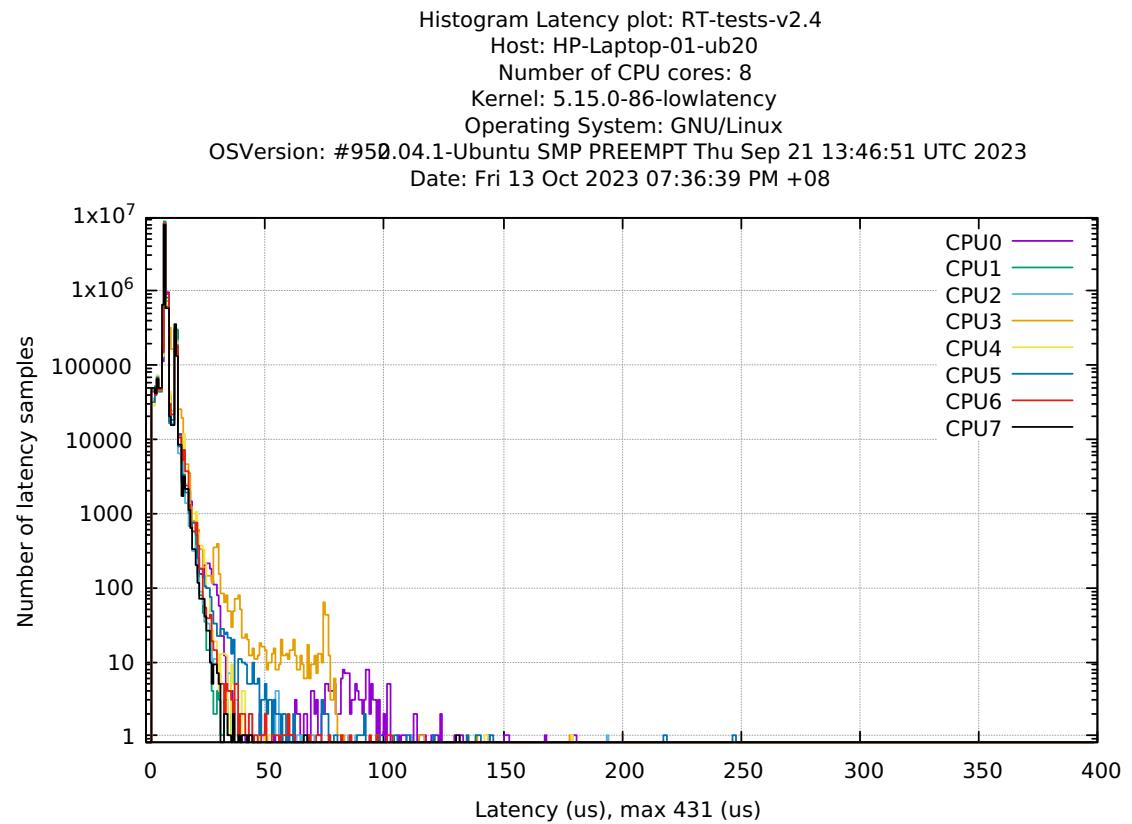


Figure 3: Histogram-Latency-HP-Laptop-02-Debian10-10million-cycles

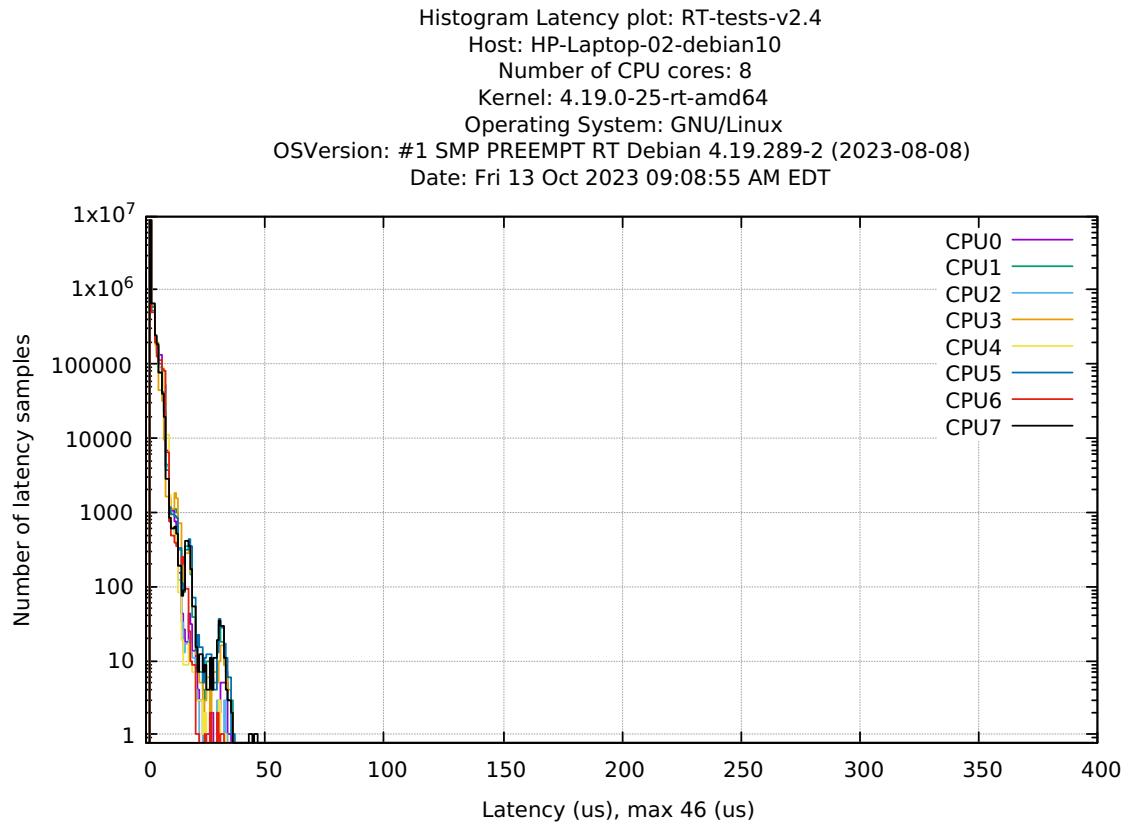
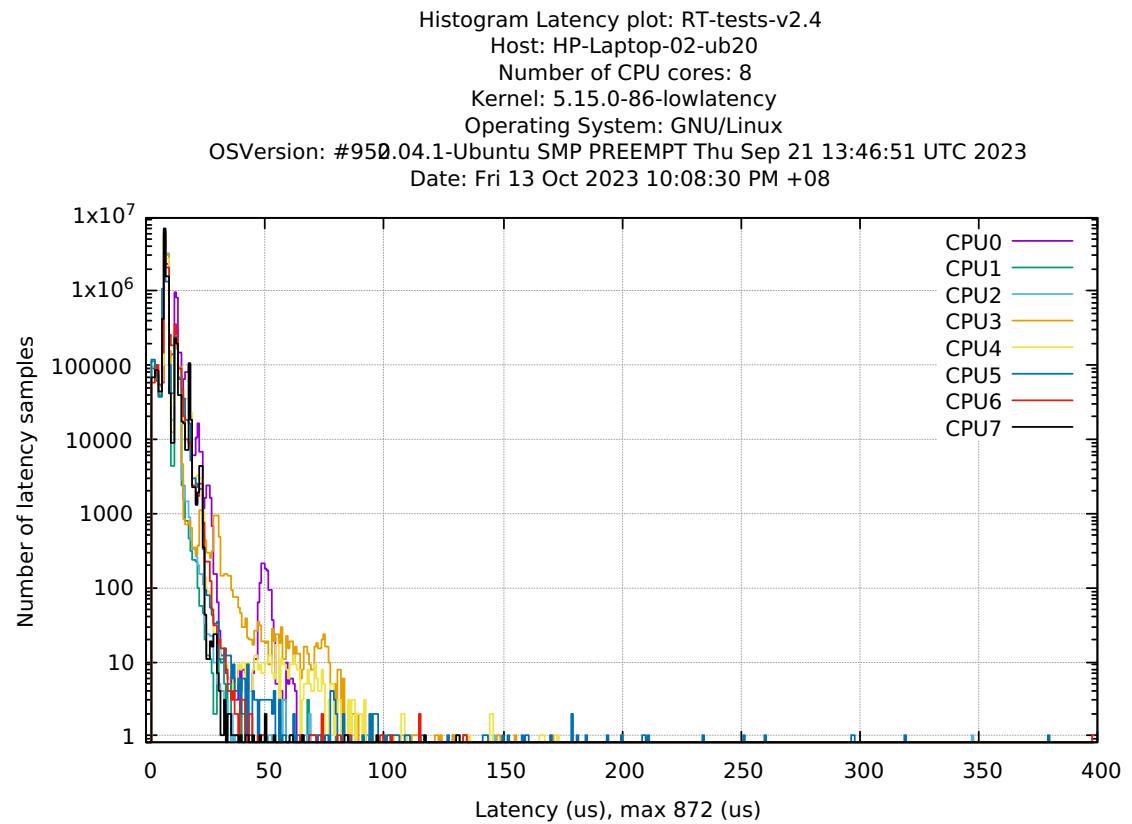


Figure 4: Histogram-Latency-HP-Laptop-02-Ubuntu20-10million-cycles



1.4 CPU Latency Performance Summary

Listing 1: Debian10 HP-Laptop-01 CPU Performance

```
# Run for 10,000,000 cycles (approx 30 minutes)
# LATENCY (us)      CPU0  CPU1  CPU2  CPU3  CPU4  CPU5  CPU6  CPU7
# Min Latencies: 00002 00002 00002 00002 00002 00002 00002 00002
# Avg Latencies: 00002 00002 00002 00002 00002 00002 00002 00002
# Max Latencies: 00021 00020 00031 00042 00016 00020 00020 00019
# Histogram Overflows: 00000 00000 00000 00000 00000 00000 00000 00000
```

Listing 2: Ubuntu20 HP-Laptop-01 CPU Performance

```
# Run for 10,000,000 cycles (approx 30 minutes)
# LATENCY (us)      CPU0  CPU1  CPU2  CPU3  CPU4  CPU5  CPU6  CPU7
# Min Latencies: 00003 00003 00003 00003 00003 00003 00003 00003
# Avg Latencies: 00008 00008 00008 00008 00008 00008 00008 00008
# Max Latencies: 00431 00040 00194 00178 00179 00247 00103 00131
# Histogram Overflows: 00001 00000 00000 00000 00000 00000 00000 00000
```

Listing 3: Debian10 HP-Laptop-02 CPU Performance

```
# Run for 10,000,000 cycles (approx 30 minutes)
# LATENCY (us)      CPU0  CPU1  CPU2  CPU3  CPU4  CPU5  CPU6  CPU7
# Min Latencies: 00002 00002 00002 00002 00002 00002 00002 00002
# Avg Latencies: 00002 00002 00002 00002 00002 00002 00002 00002
# Max Latencies: 00035 00037 00034 00036 00033 00037 00032 00046
# Histogram Overflows: 00000 00000 00000 00000 00000 00000 00000 00000
```

Listing 4: Ubuntu20 HP-Laptop-02 CPU Performance

```
# Run for 10,000,000 cycles (approx 30 minutes)
# LATENCY (us)      CPU0  CPU1  CPU2  CPU3  CPU4  CPU5  CPU6  CPU7
# Min Latencies: 00003 00003 00003 00003 00003 00003 00003 00003
# Avg Latencies: 00009 00008 00008 00008 00008 00008 00008 00008
# Max Latencies: 00065 00093 00872 00200 00173 00379 00398 00131
# Histogram Overflows: 00000 00000 00002 00000 00000 00000 00000 00000
```

.2 APPENDIX CHAPTER 4

.2.1 Ellipse Perimeter using Ramanujan Approximation

Ramanujan is a well known mathematician. We used Ramanujan approximation formula to calculate the perimeter of an Ellipse. The reference is the website at URL [Perimeter of an Ellipse]

The perimeter calculation equation for the Ellipse is given by Ramanujan as follows:

$Perimeter = PI * (a + b) * (1 + C/D)$ where a and b are the semi-major and semi-minor lengths, respectively. C and D are based on another variable h, defined as follows.

Introduce $h = (a - b)^2/(a + b)^2$ we get

$$C = 3 * h \text{ and } D = 10 + \sqrt{4 - 3 * h}$$

For a specific Ellipse with $a = 51$ and $b = 11$, we get

$$h = (a - b)^2/(a + b)^2$$

$$h = (40)^2/(62)^2 = (1600)/(3844)$$

$$h = 0.416233090530697$$

With h, the bvalue for C is:

$$C = 3 * h = 3 * (0.416233090530697)$$

$$C = 1.24869927159209$$

and the value for D is:

$$D = 10 + \sqrt{4 - 3 * h}$$

$$D = 10 + \sqrt{4 - 1.24869927159209}$$

$$D = 10 + \sqrt{2.75130072840791}$$

$$D = 10 + 1.65870453318483$$

$$D = 11.65870453318483$$

The formula for Ramanujan approximation of the perimeter on an Ellipse:

$$\text{Perimeter} = \text{PI} * (a + b) * (1 + C/D)$$

$$(C/D) = (1.24869927159209 / 11.65870453318483)$$

$$(C/D) = 0.107104461566707$$

$$\text{Perimeter} = \text{PI} * (a + b) * (1 + C/D)$$

$$\text{Perimeter} = \text{PI} * (62) * (1 + 0.107104461566707)$$

$$\text{Perimeter} = \text{PI} * (62) * (1.107104461566707)$$

Using PI = 3.141592653589793238

$$\text{Perimeter} = 215.640417079296$$

Perimeter = 2.156404170793E + 02 as calculated using the Ramanujan approximation formula.

Perimeter = 2.156436635306E + 02 as calculated by the parametric curve interpolation

algorithm in this work for comparison.

2.2 Calculation of machine epsilon C code

Listing 5: Calculation of machine epsilon C code

```
// Calculation of machine epsilon
# include <stdio.h>
int main(void) {

    float      floatMache = 1.0;
    double     doubleMache = 1.0;
    long double longDblMache = 1.0;

    while (1.0 + floatMache / 2.0 != 1.0) {
        floatMache /= 2.0;
    }
    while (1.0 + doubleMache / 2.0 != 1.0) {
        doubleMache /= 2.0;
    }
    while (1.0 + longDblMache / 2.0 != 1.0) {
        longDblMache /= 2.0;
    }

    printf("Machine Epsilon float type\t= %.19e\n", floatMache);
    printf("Machine Epsilon double type\t= %.19e\n", doubleMache);
    printf("Machine Epsilon long double type\t= %.19Le\n", longDblMache);

    return (0);
}
/*
COMPILATION
wruslan@HP-Laptop-01:~$ gcc -o calculate-mache .c calculate-mache.c

EXECUTION
wruslan@HP-Laptop-01:~$ ./calculate-mache
Machine Epsilon float type      = 2.2204460492503130808e-16
Machine Epsilon double type     = 2.2204460492503130808e-16
Machine Epsilon long double type = 1.0842021724855044340e-19
wruslan@HP-Laptop-01:~$ */
```

.2.3 Ranges of floating-point numbers in C code

Listing 6: Display ranges of floating-point and integer numbers

2.4 Machine epsilon affecting addition and subtraction operations

Listing 7: Machine epsilon affecting addition and subtraction operations

```
// Machine epsilon affecting addition and subtraction operations
# include <stdio.h>
# include <limits.h>
# include <float.h>
int main(void) {
    double largeDouble = 3.123456789E+6;
    double smallDouble = 1.123456789E-18;           // BELOW MACHINE EPSILON
    printf("\n(1) ADDITION OF SMALL NUMBER BELOW MACHINE EPSILON \n");
    printf(" largeDouble %.12e \n", largeDouble);
    printf("+ smallDouble %.12e \n", smallDouble);
    printf("= %.12e \n", (largeDouble + smallDouble));
    printf("\n(2) SUBTRACTION OF SMALL NUMBER BELOW MACHINE EPSILON \n");
    printf(" largeDouble %.12e \n", largeDouble);
    printf("- smallDouble %.12e \n", smallDouble);
    printf("= %.12e \n", (largeDouble - smallDouble));
    printf("\n(3) MULTIPLICATION BY SMALL NUMBER BELOW MACHINE EPSILON\n");
    printf(" largeDouble %.12e \n", largeDouble);
    printf("* smallDouble %.12e \n", smallDouble);
    printf("= %.12e \n", (largeDouble * smallDouble));
    printf("\n(4) DIVISION BY SMALL NUMBER BELOW MACHINE EPSILON \n");
    printf(" largeDouble %.12e \n", largeDouble);
    printf("/ smallDouble %.12e \n", smallDouble);
    printf("= %.12e \n", (largeDouble / smallDouble));
    return (0);
}
/*
COMPILATION
wruslan@HP-Laptop-01:~$ gcc -o fixing-machine-epsilon.c fixing-machine-epsilon.c
EXECUTION
wruslan@HP-Laptop-01:~$ ./fixing-machine-epsilon.c

(1) ADDITION OF SMALL NUMBER BELOW MACHINE EPSILON
 largeDouble 3.123456789000e+06
+ smallDouble 1.123456789000e-18
=
= 3.123456789000e+06

(2) SUBTRACTION OF SMALL NUMBER BELOW MACHINE EPSILON
 largeDouble 3.123456789000e+06
- smallDouble 1.123456789000e-18
=
= 3.123456789000e+06

(3) MULTIPLICATION BY SMALL NUMBER BELOW MACHINE EPSILON
 largeDouble 3.123456789000e+06
* smallDouble 1.123456789000e-18
=
= 3.509068734750e-12

(4) DIVISION BY SMALL NUMBER BELOW MACHINE EPSILON
 largeDouble 3.123456789000e+06
/ smallDouble 1.123456789000e-18
=
= 2.780219782000e+24
*/
wruslan@HP-Laptop-01:~$
```

2.5 Resolving machine epsilon issue

Listing 8: Resolving machine epsilon issue

```
# include <stdio.h>
# include <limits.h>
# include <float.h>
int main(void) {

    double largeDouble = 3.123456789E+6;
    double smallDouble = 1.123456789E-18;      // BELOW MACHINE EPSILON
    double machepFactor = 1.8765E+10;          // A LARGE POSITIVE NUMBER

    printf("ADDITION OF SMALL NUMBER BELOW MACHINE EPSILON \n");
    printf(" largeDouble %.12e \n", largeDouble);
    printf("+ smallDouble %.12e \n", smallDouble);
    printf("= %.12e \n", (largeDouble + smallDouble));

    double uplargeDouble = (machepFactor)*largeDouble;
    double upsmallDouble = (machepFactor)*smallDouble;

    printf("\n");
    printf("uplargeDouble %.12e \n", uplargeDouble);
    printf("upsmallDouble %.12e \n", upsmallDouble);

    double the_SUM_01 = (uplargeDouble + upsmallDouble);
    double the_SUM_02 = (the_SUM_01)/(machepFactor);

    printf("the_SUM_01 = %.12e \n", the_SUM_01);
    printf("the_SUM_02 = %.12e \n", the_SUM_02);

    return (0);
}
/*
COMPILATION
wruslan@HP-Laptop -01:~$ gcc -o Resolving-machine-epsilon.c Resolving-machine-epsilon.c

EXECUTION
wruslan@HP-Laptop -01:~$ ./Resolving-machine-epsilon.c

ADDITION OF SMALL NUMBER BELOW MACHINE EPSILON
    largeDouble 3.123456789000e+06
+ smallDouble 1.123456789000e-18
=
            3.123456789000e+06

    uplargeDouble 5.861166664558e+16
    upsmallDouble 2.108166664558e-08
    the_SUM_01 = 5.861166664558e+16
    the_SUM_02 = 3.123456789000e+06

wruslan@HP-Laptop -01:~$ */

```

.2.6 LinuxCNC Validation of Curves

The following list provides the links to the screen capture of real runs of the G-codes generated by the interpolation algorithm on the LinuxCNC-Axis control computer. The computer drives a CNC machine by sending electrical signals via the standard parallel port. The figures in the next ten(10) pages are provided in landscape mode.

1. Circle curve validation link [5]
2. Ellipse curve validation link [6]
3. Teardrop curve validation link [7]
4. Butterfly curve validation link [8]
5. Snailshell curve validation link [9]
6. Skewed-Astroid curve validation link [10]
7. Ribbon-10L curve validation link [11]
8. Ribbon-100L curve validation link [12]
9. AstEpi curve validation link [13]
10. SnaHyp curve validation link [14]

Figure 5: Circle validation LinuxCNC-Axis execution

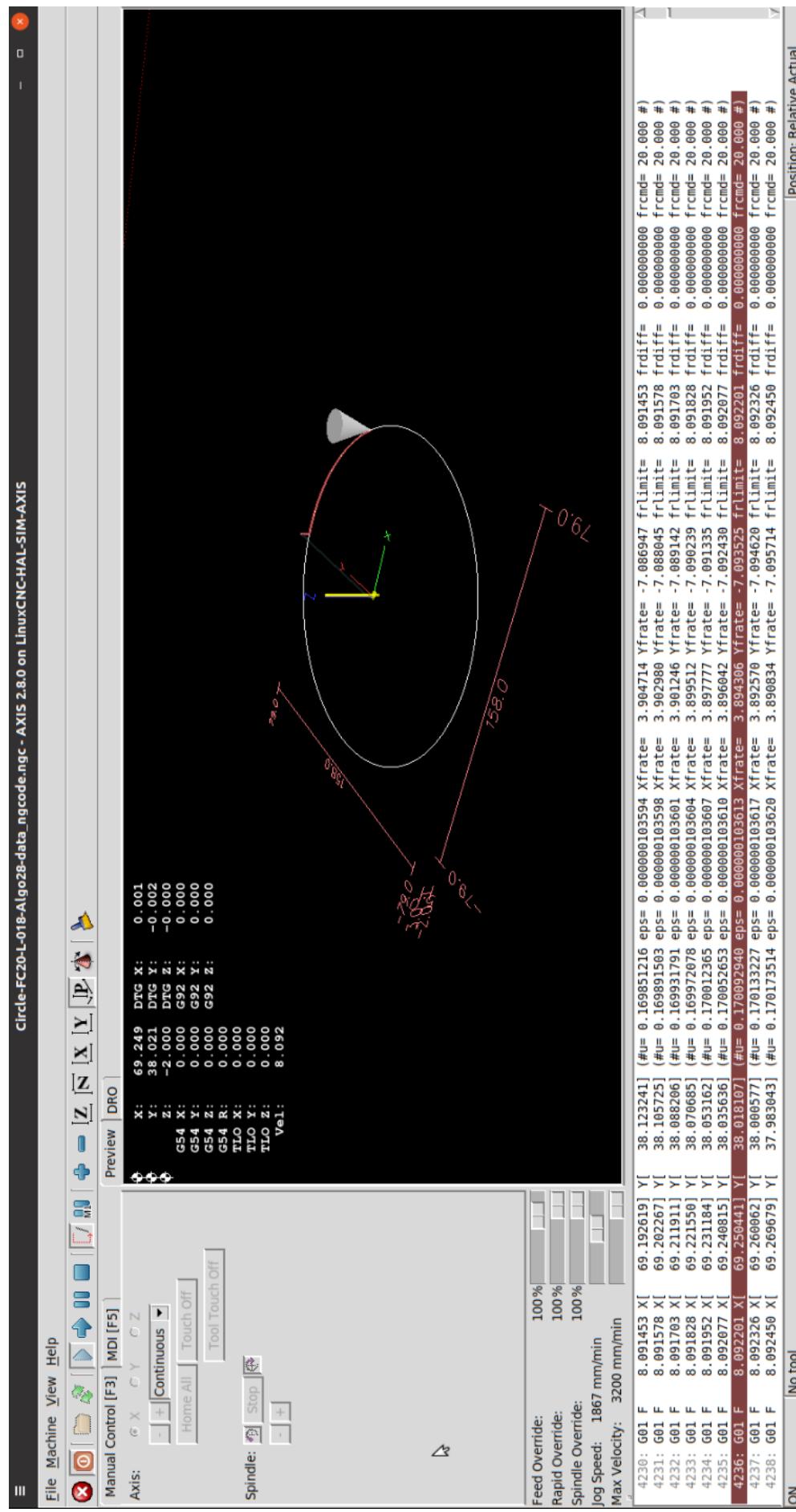


Figure 6: Ellipse validation LinuxCNC-Axis execution

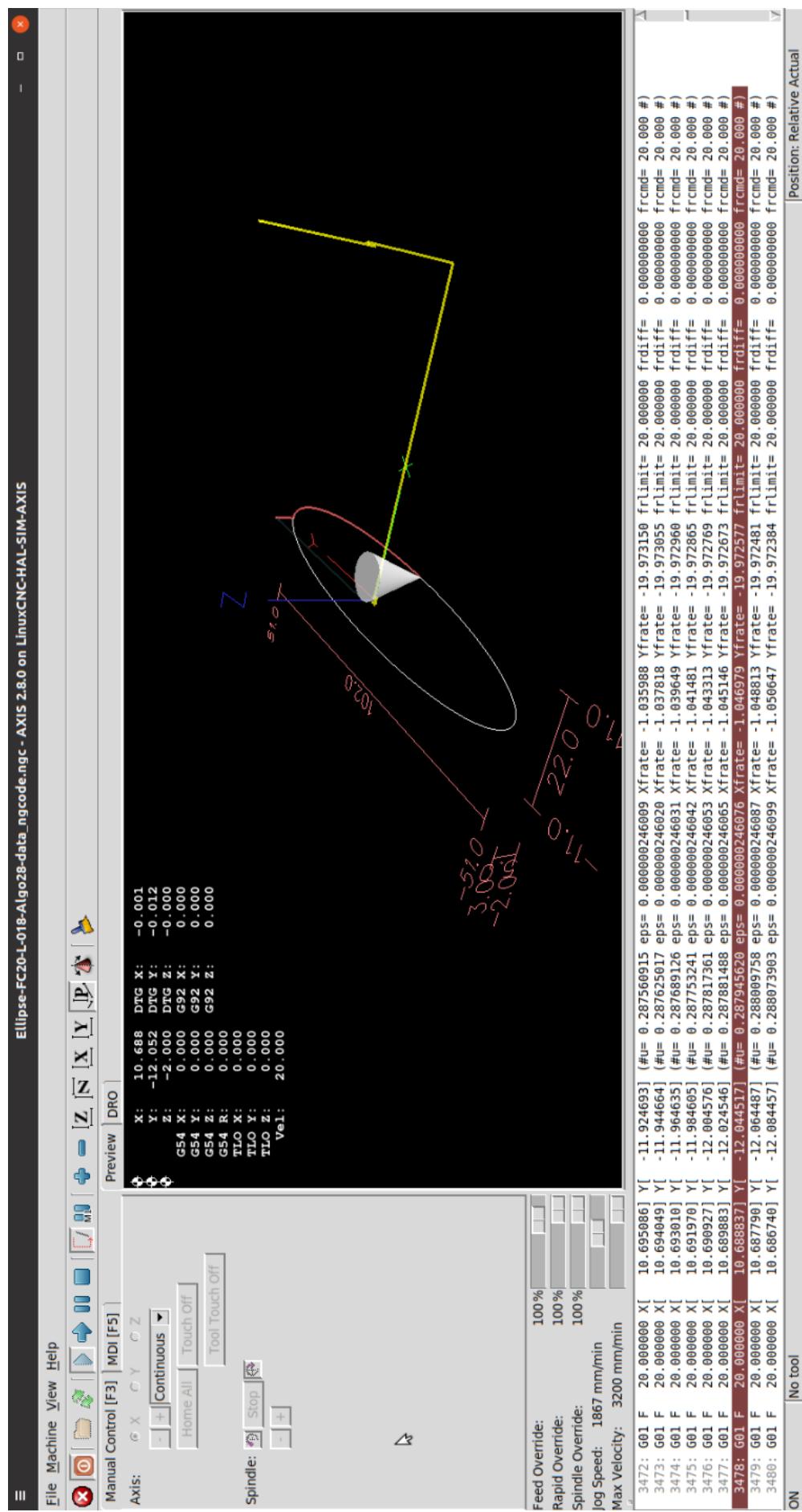


Figure 7: Teardrop validation LinuxCNC-Axis execution

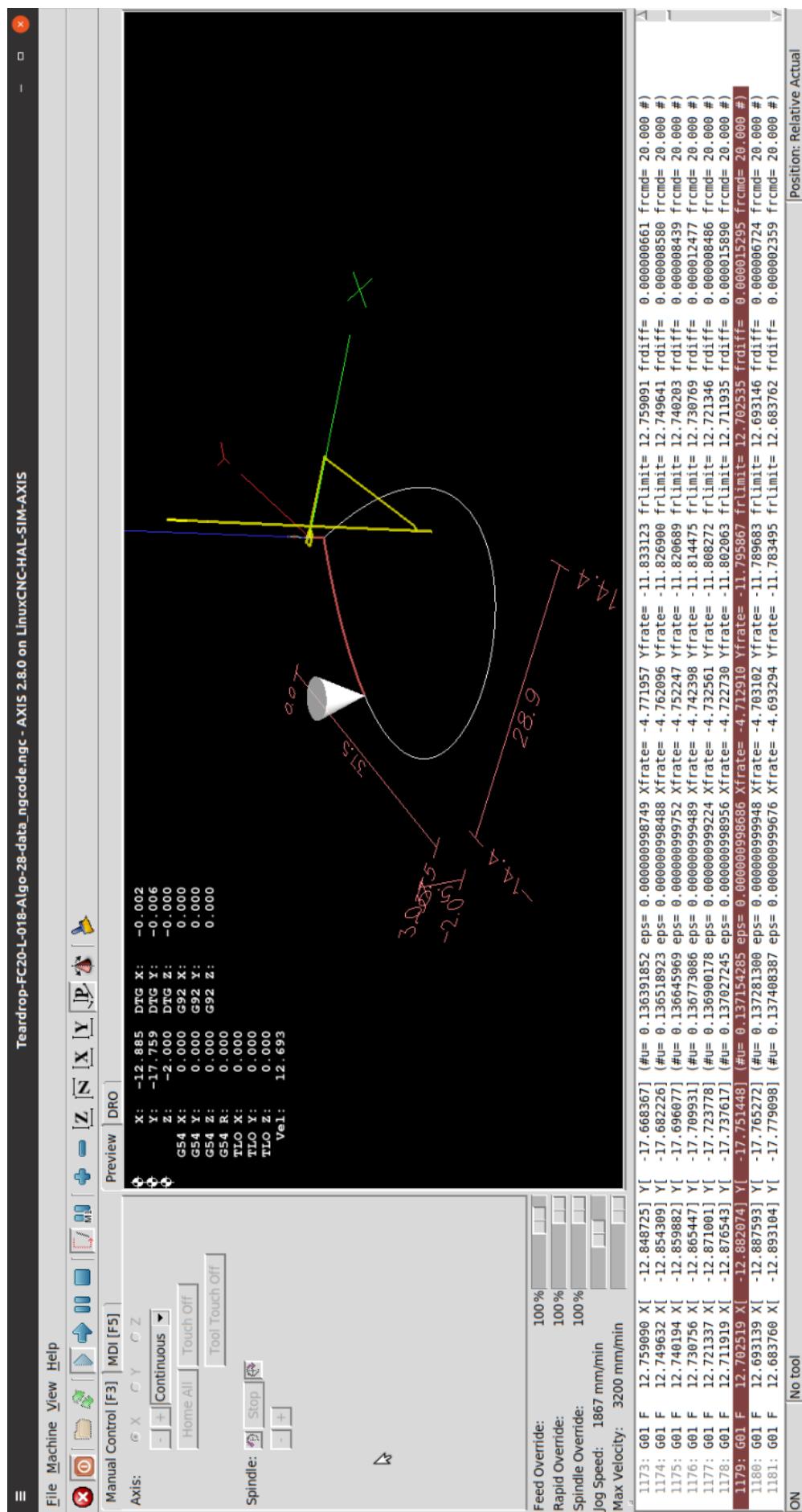


Figure 8: Butterfly validation LinuxCNC-Axis execution

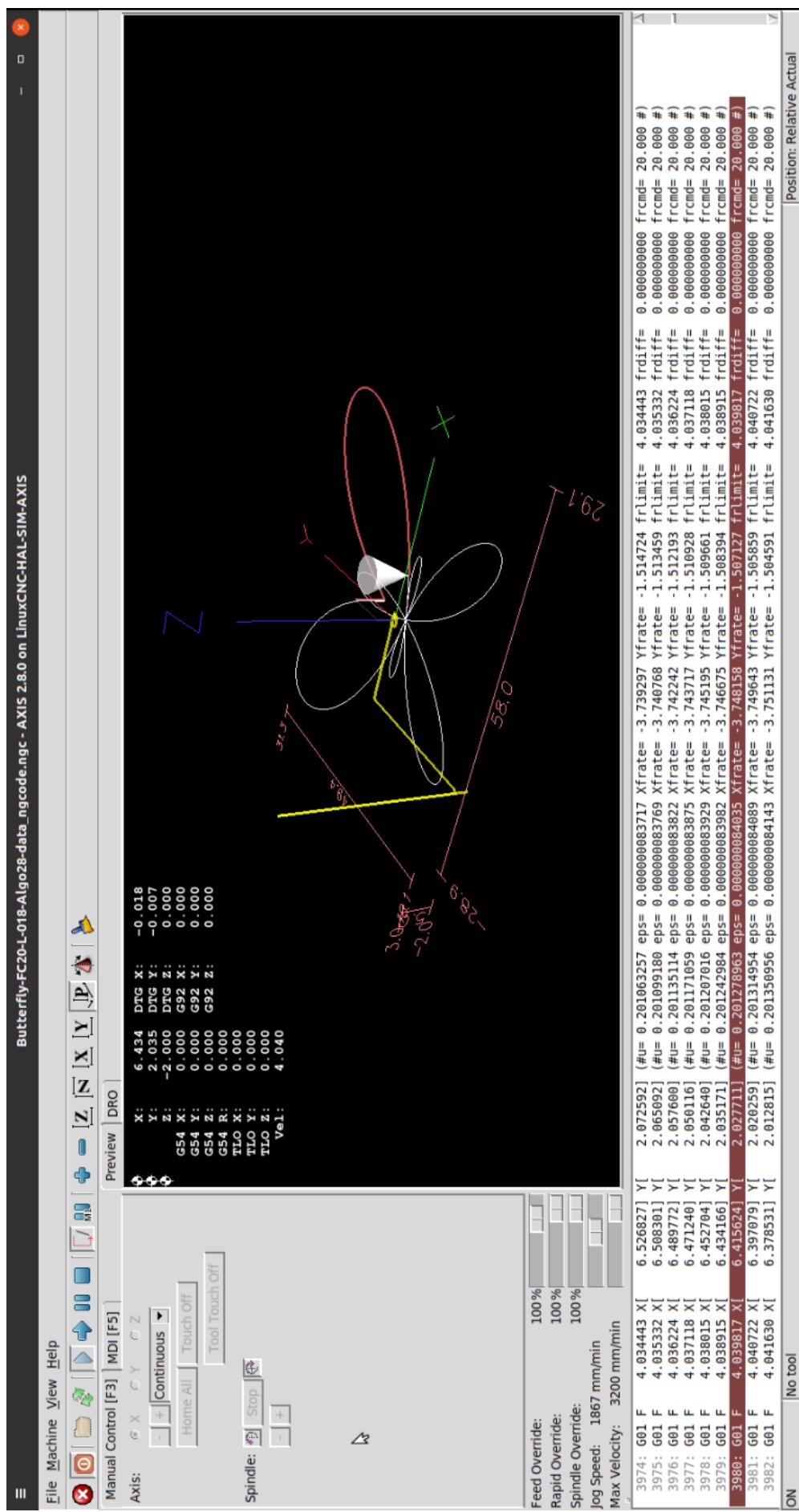


Figure 9: Snailshell validation LinuxCNC-Axis execution

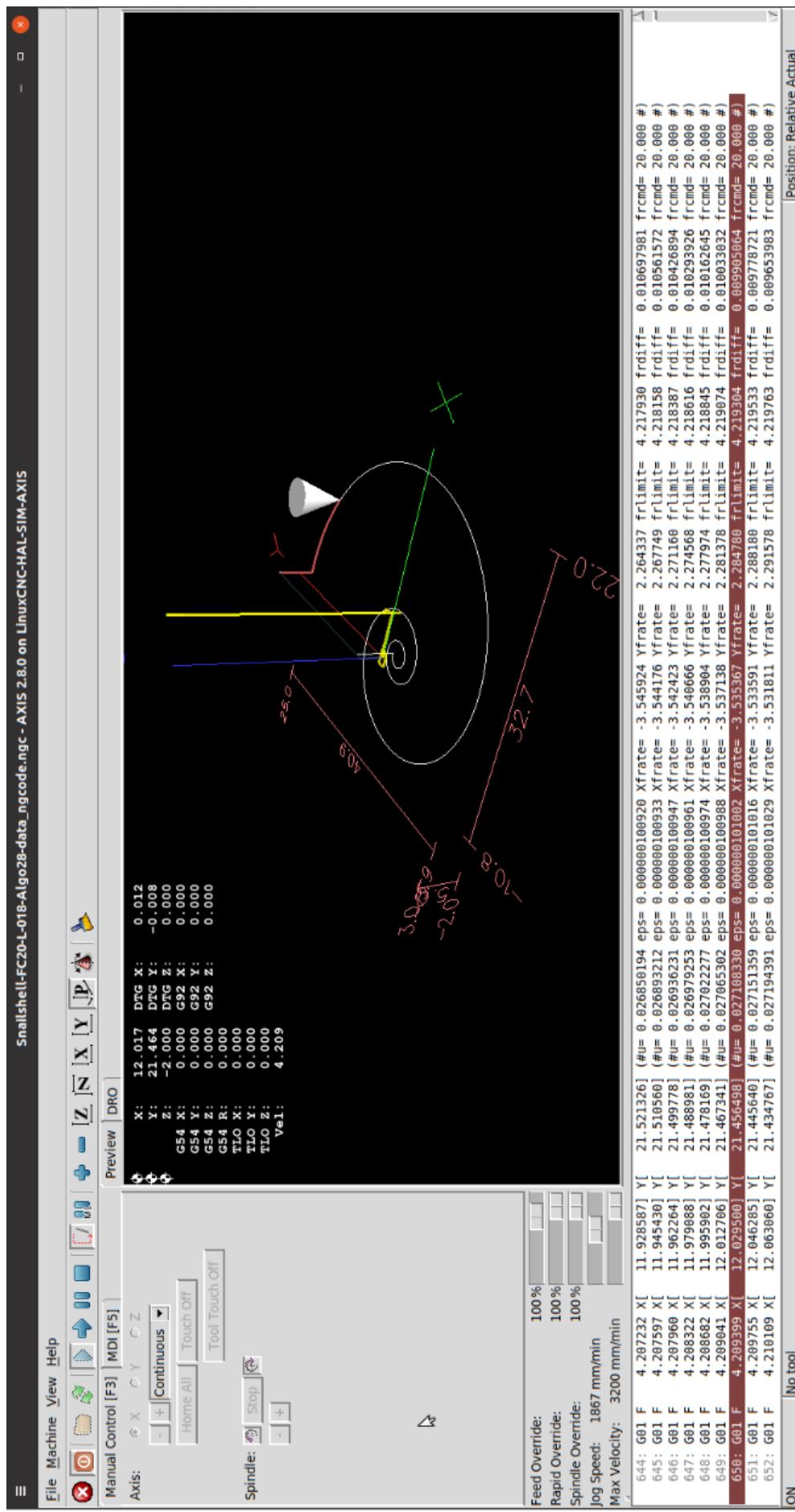


Figure 10: Skewed-Astroid validation LinuxCNC-Axis execution

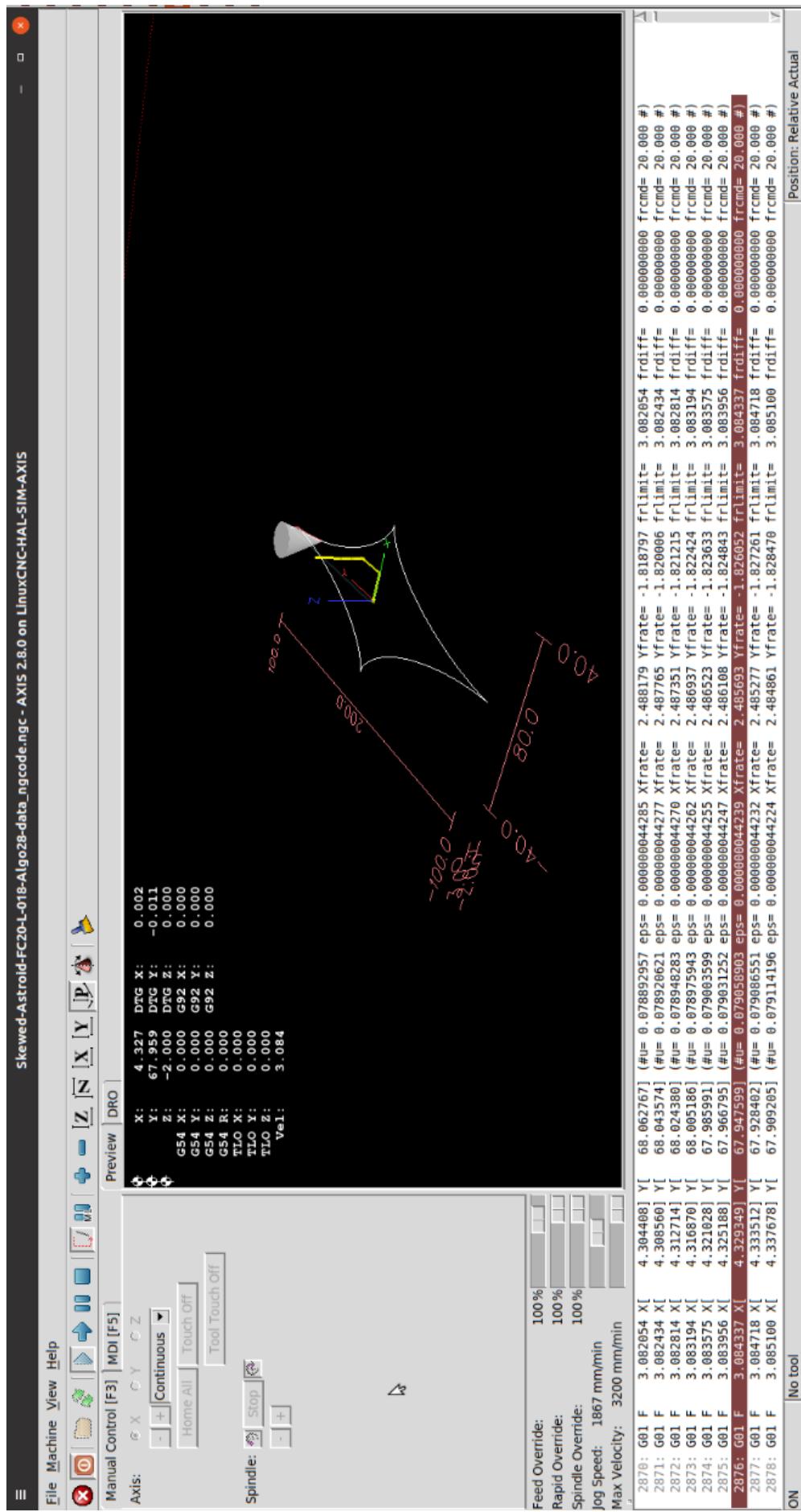


Figure 11: Ribbon-10L validation LinuxCNC-Axis execution

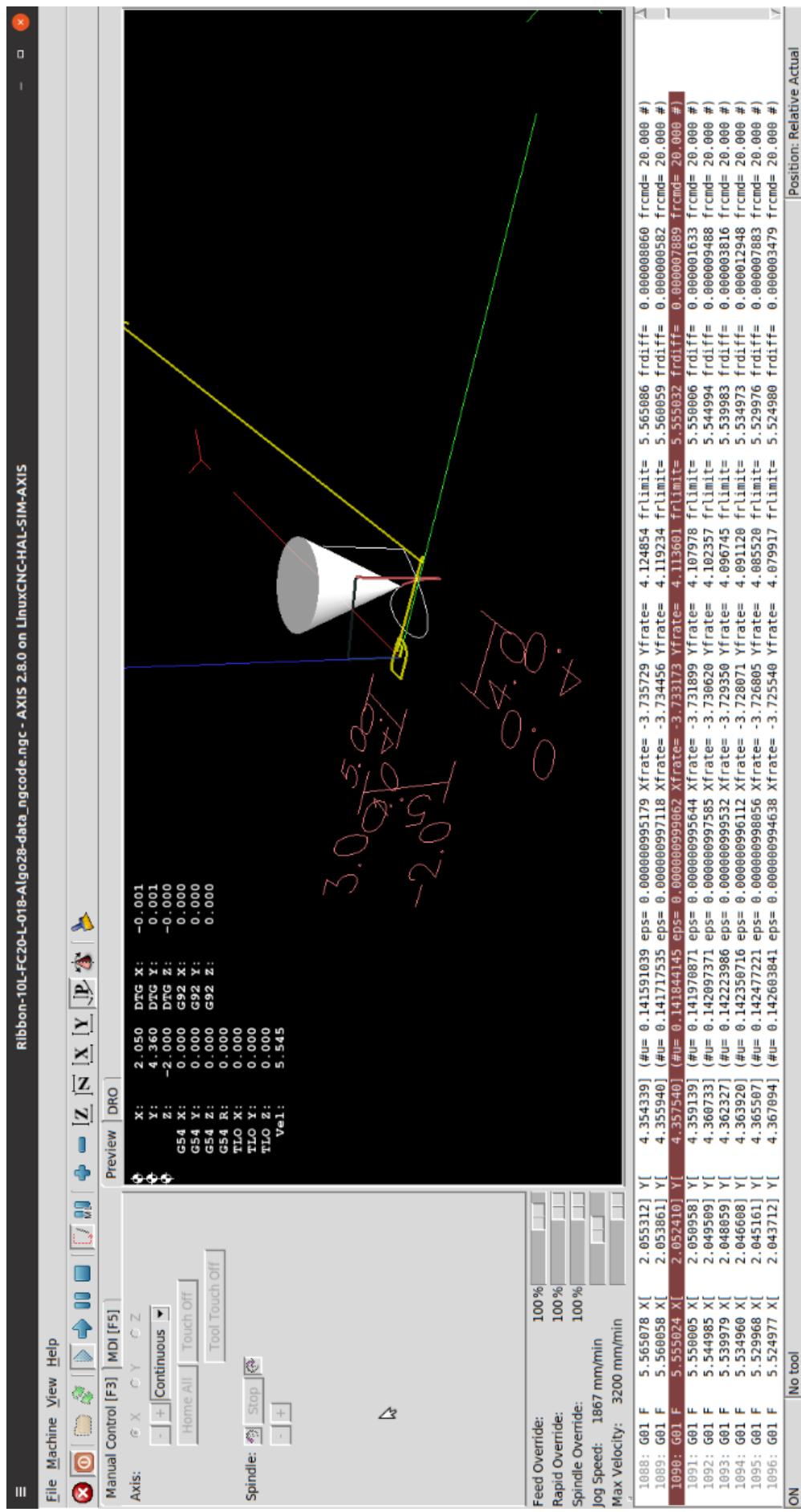


Figure 12: Ribbon-100L validation LinuxCNC-Axis execution

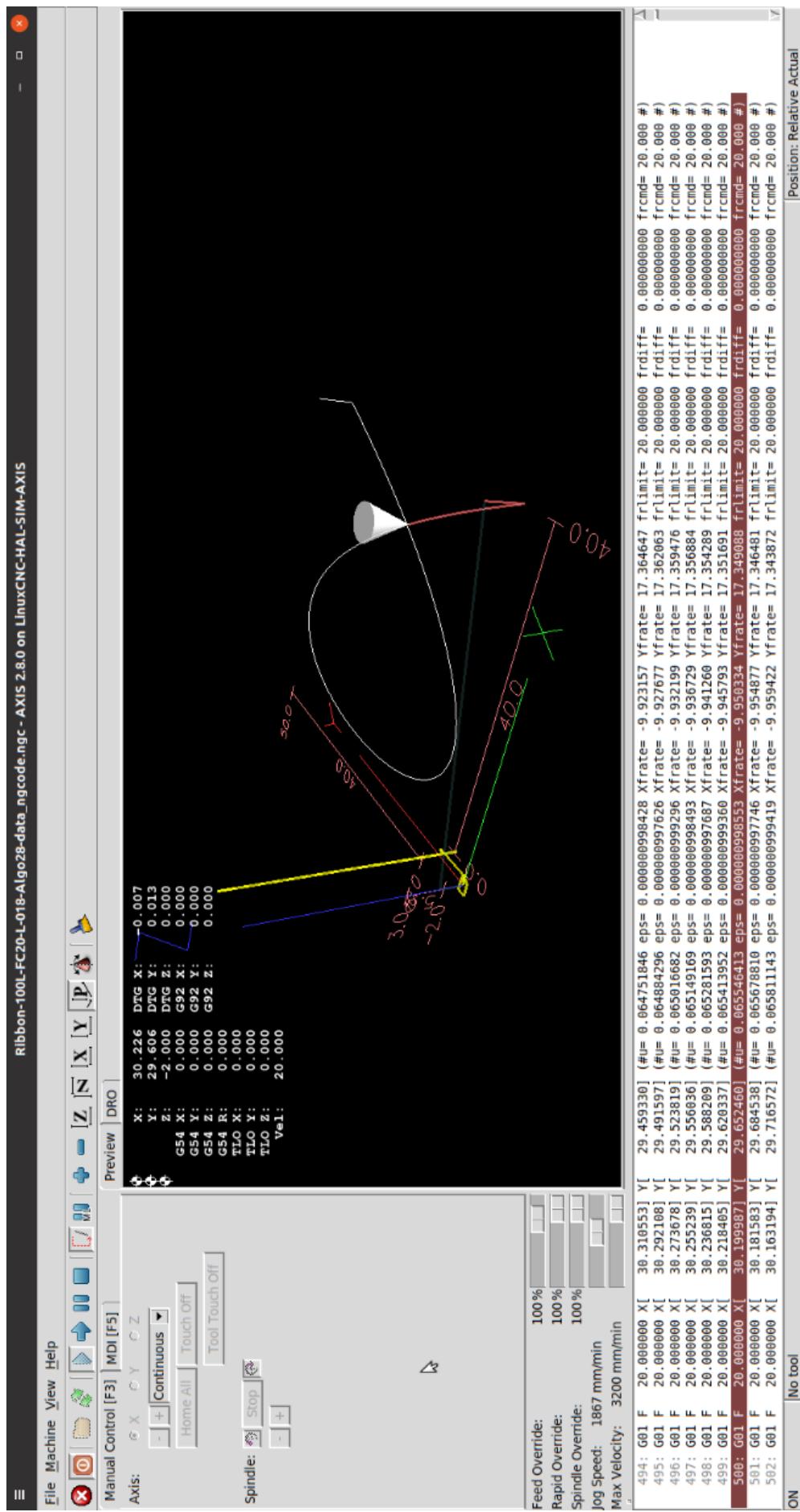


Figure 13: AstEpi validation LinuxCNC-Axis execution

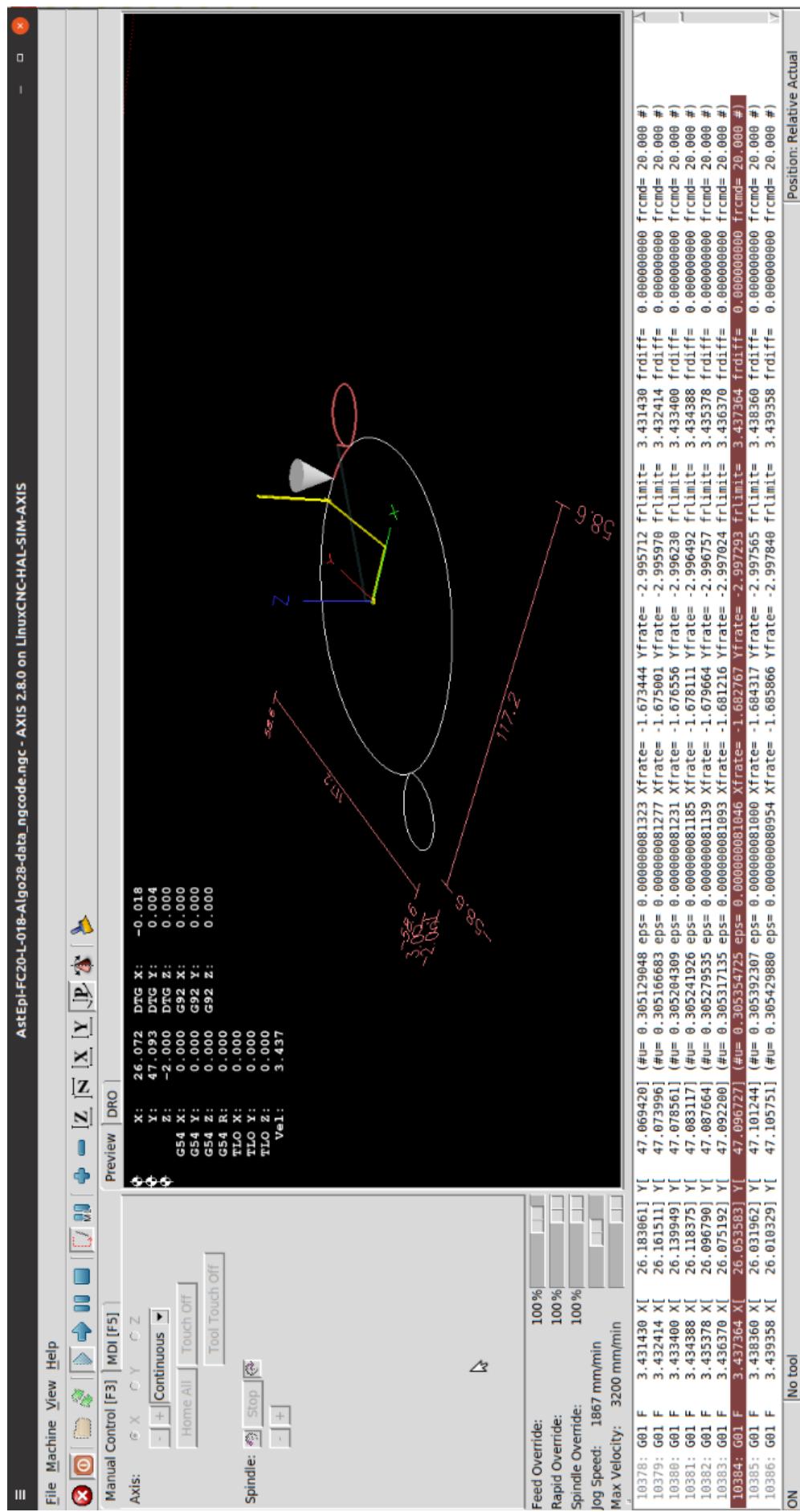
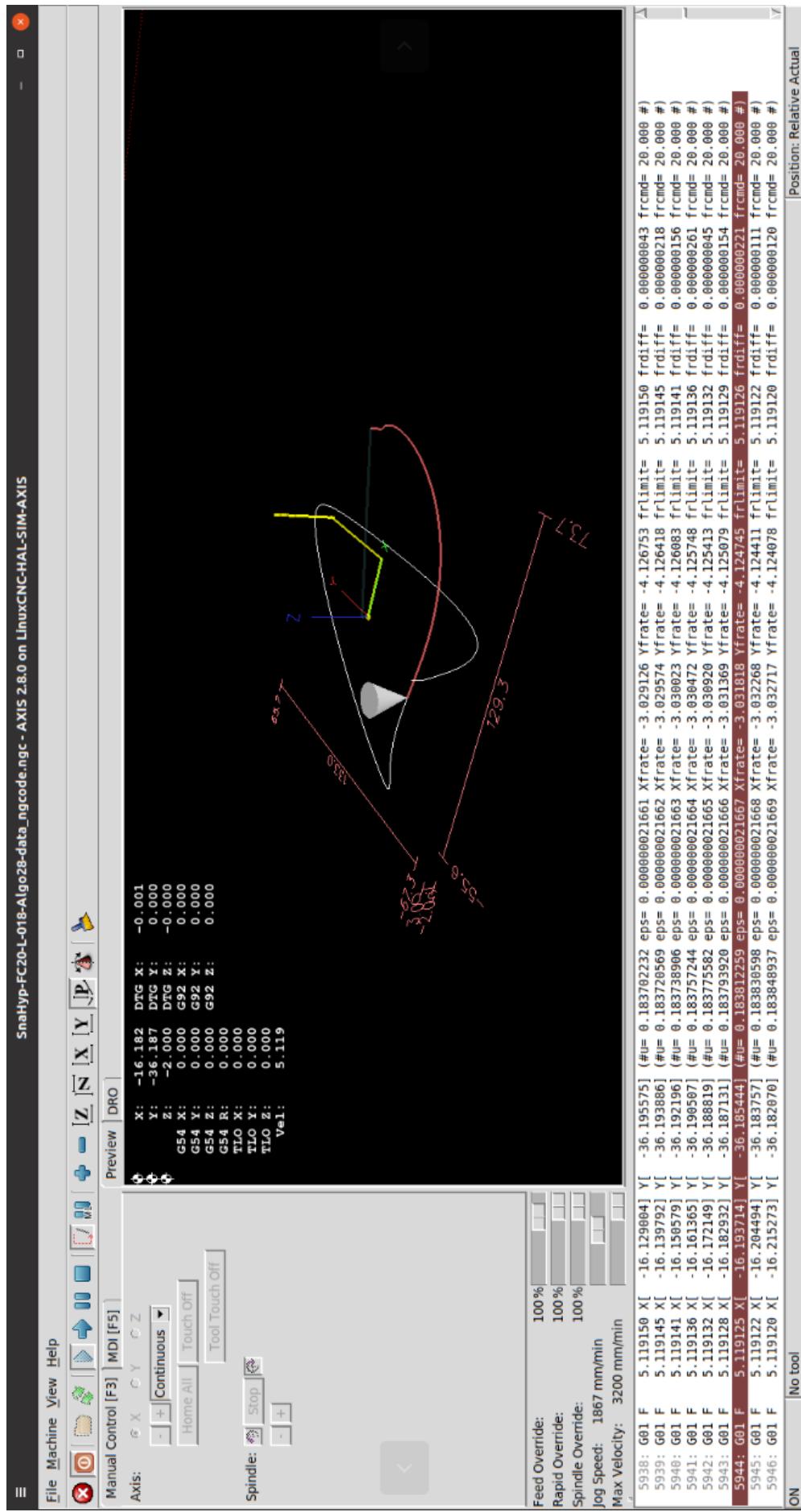


Figure 14: SnaHyp validation LinuxCNC-Axis execution



.3 APPENDIX CIRCLE CURVE

- .3.1 Plot of Circle curve [15]**
- .3.2 Circle Radius of Curvature [16]**
- .3.3 Circle Validation in LinuxCNC [17]**
- .3.4 Circle Direction of Travel 3D [18]**
- .3.5 Circle First and Second Order Taylor's Approx [19]**
- .3.6 Circle First minus Second Order Taylor's Approx [20]**
- .3.7 Circle Separate First and Second Order Taylor's Approx [21]**
- .3.8 Circle Separation SAL and SCL [22]**
- .3.9 Circle Chord-error in close view 2 scales [23]**
- .3.10 Circle Four Components Feedrate Limit [24]**
- .3.11 Circle FrateCommand FrateLimit and Curr-Frate [25]**
- .3.12 Circle FeedRateLimit minus CurrFeedRate [26]**
- .3.13 Circle FC20-Nominal X and Y Feedrate Profiles [27]**
- .3.14 Circle FC20 Nominal Tangential Acceleration [28]**
- .3.15 Circle FC20 Nominal Rising S-Curve Profile [29]**
- .3.16 Circle FC20 Nominal Falling S-Curve Profile [30]**
- .3.17 Circle FC10 Colored Feedrate Profile data ngcode [31]**
- .3.18 Circle FC20 Colored Feedrate Profile data ngcode [32]**

- .3.19 Circle FC30 Colored Feedrate Profile data ngcode [33]
- .3.20 Circle FC40 Colored Feedrate Profile data ngcode [34]
- .3.21 Circle FC10 Tangential Acceleration [35]
- .3.22 Circle FC20 Tangential Acceleration [36]
- .3.23 Circle FC30 Tangential Acceleration [37]
- .3.24 Circle FC40 Tangential Acceleration [38]
- .3.25 Circle FC20 Nominal Separation NAL and NCL [39]
- .3.26 Circle SAL minus SCL for FC10 FC20 FC30 FC40 [40]
- .3.27 Circle FC10 FrateCmd CurrFrate X-Frate Y-Frate [41]
- .3.28 Circle FC20 FrateCmd CurrFrate X-Frate Y-Frate [42]
- .3.29 Circle FC30 FrateCmd CurrFrate X-Frate Y-Frate [43]
- .3.30 Circle FC40 FrateCmd CurrFrate X-Frate Y-Frate [44]
- .3.31 Circle FC10 Four Components FeedrateLimit [45]
- .3.32 Circle FC20 Four Components FeedrateLimit [46]
- .3.33 Circle FC30 Four Components FeedrateLimit [47]
- .3.34 Circle FC40 Four Components FeedrateLimit [48]
- .3.35 Circle Histogram Points FC10 FC20 FC30 FC40 [49]
- .3.36 Circle Table distribution of interpolated points [2]
- .3.37 Circle Table FC10-20-30-40 Run Performance data [3]

Figure 15: Plot of Circle curve

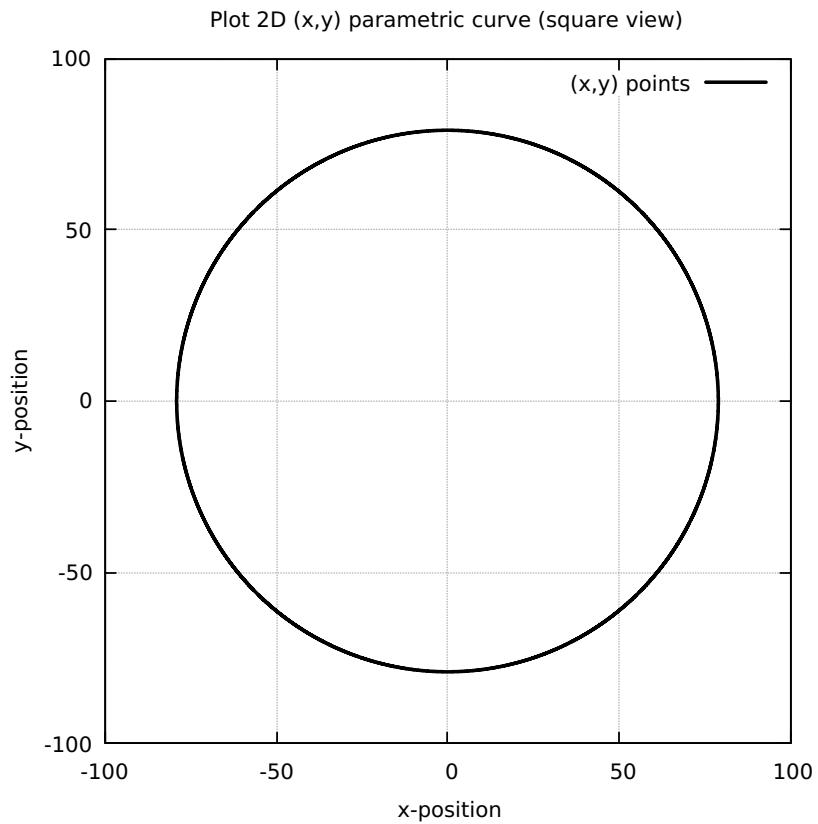


Figure 16: Circle Radius of Curvature

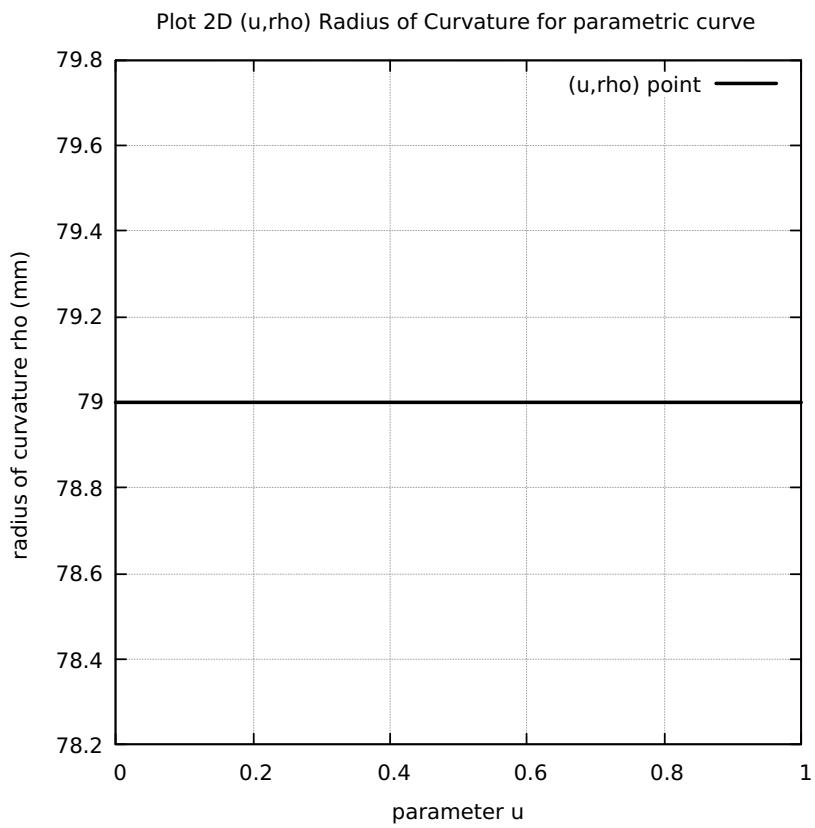


Figure 17: Circle Validation in LinuxCNC

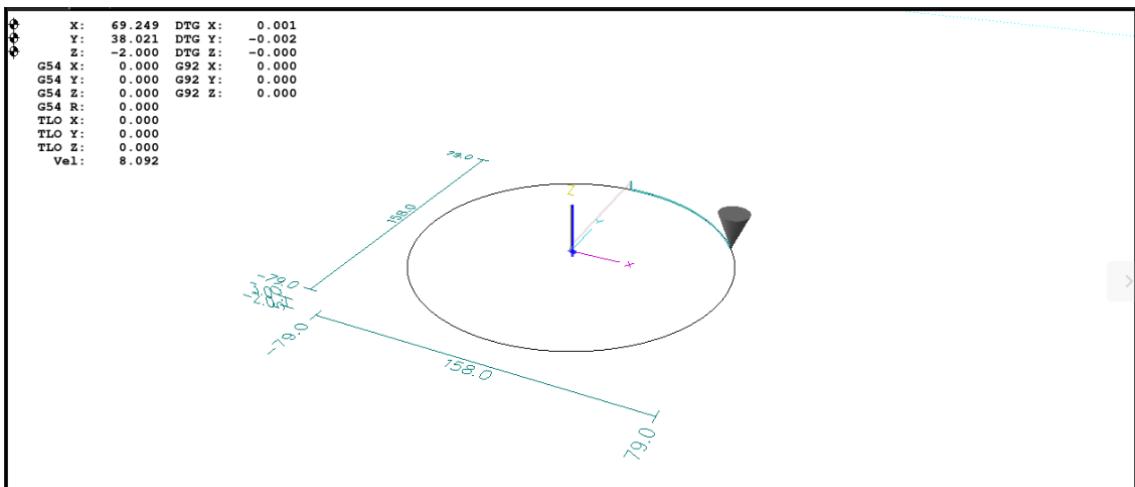


Figure 18: Circle Direction of Travel 3D

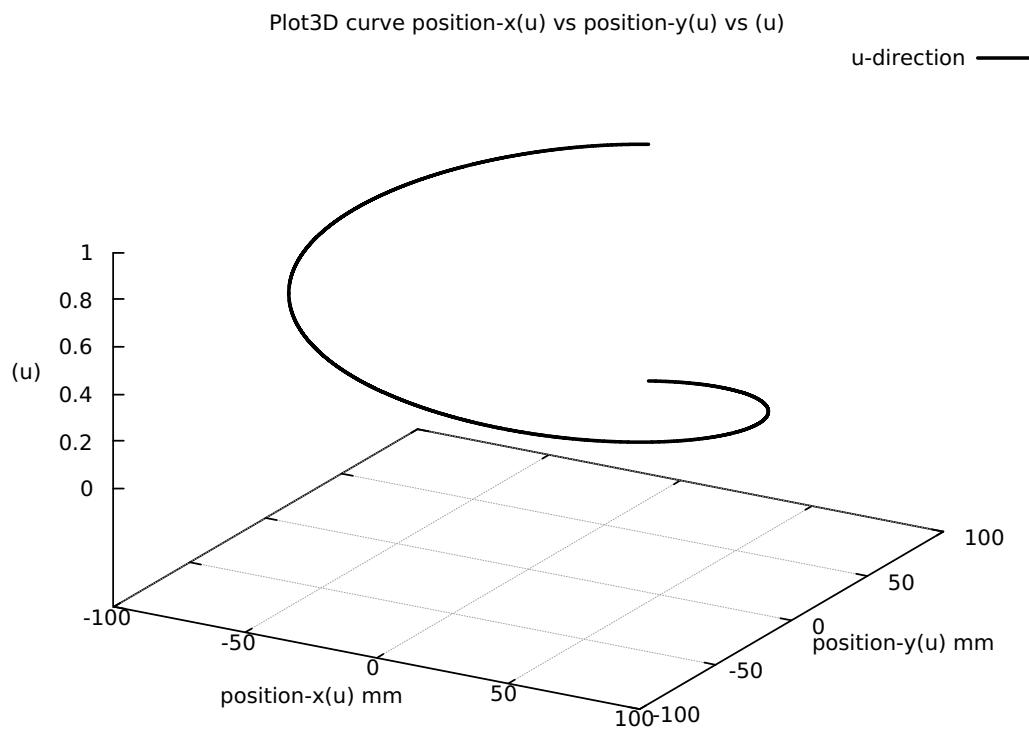


Figure 19: Circle First and Second Order Taylor's Approximation

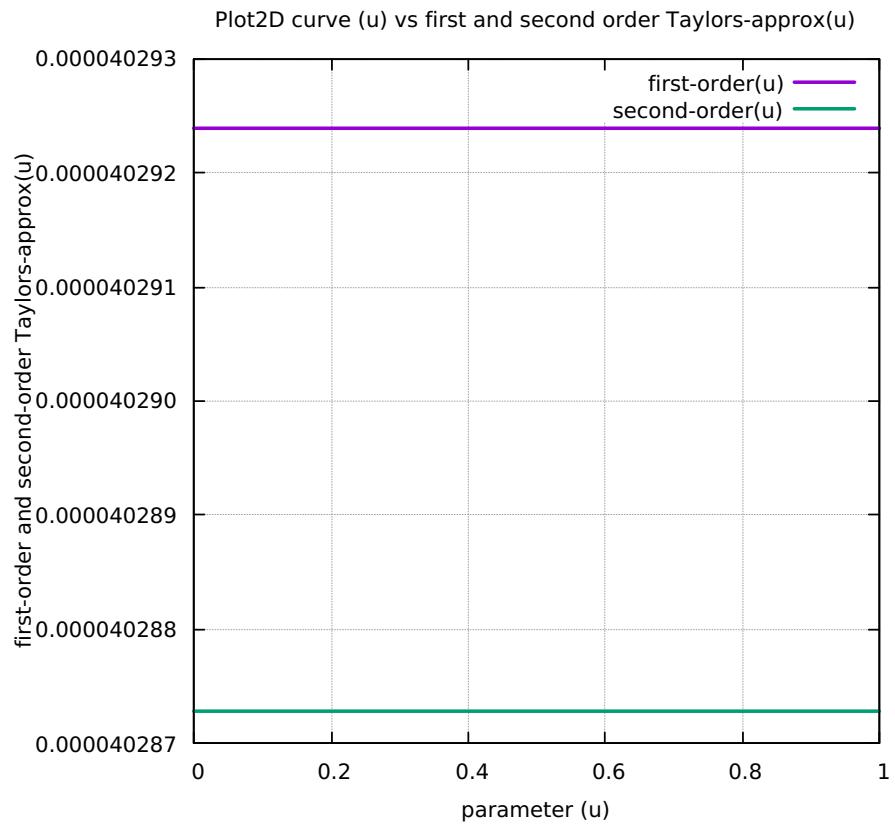


Figure 20: Circle First minus Second Order Taylor's Approximation

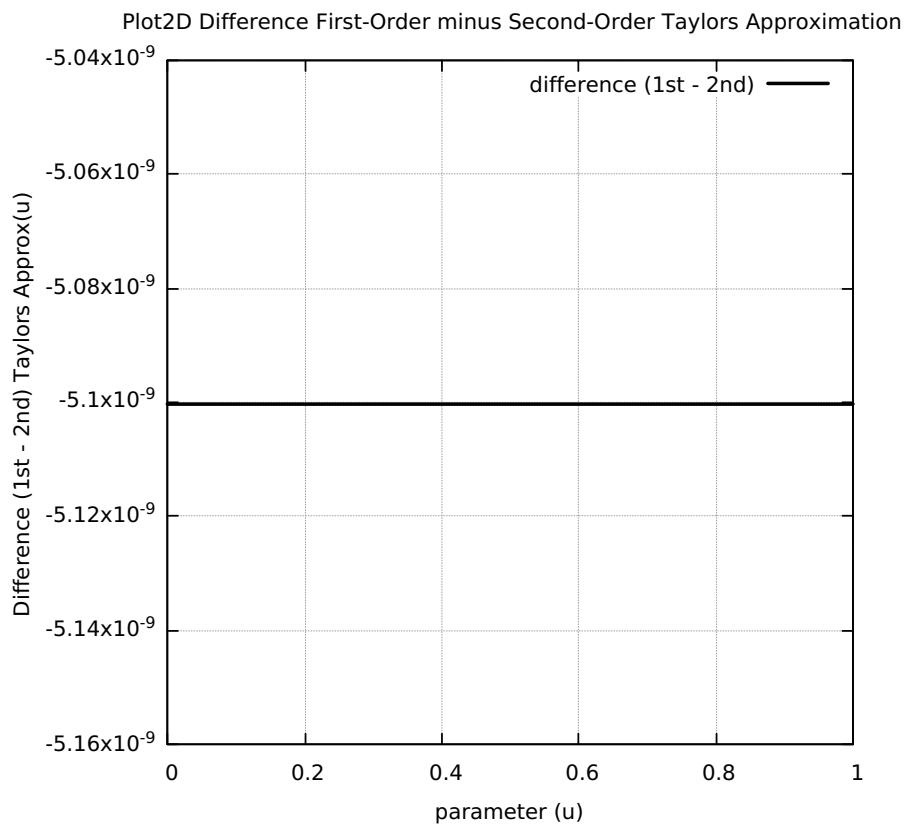


Figure 21: Circle Separation First and Second Order Taylor's Approximation

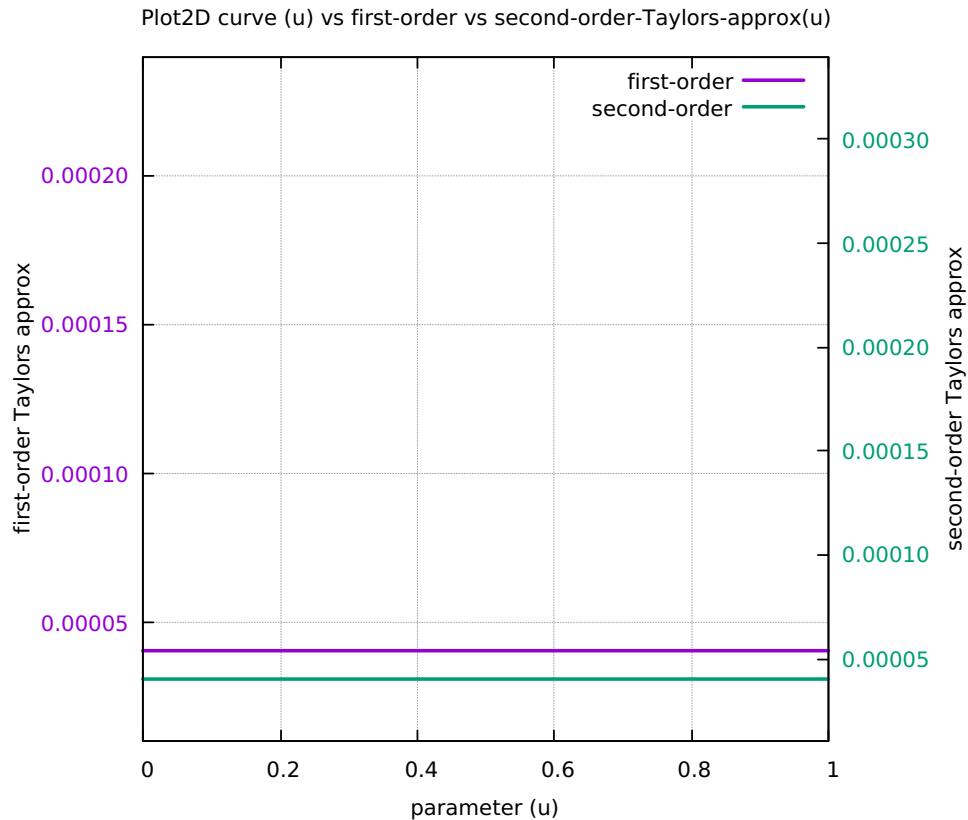


Figure 22: Circle Separation SAL and SCL

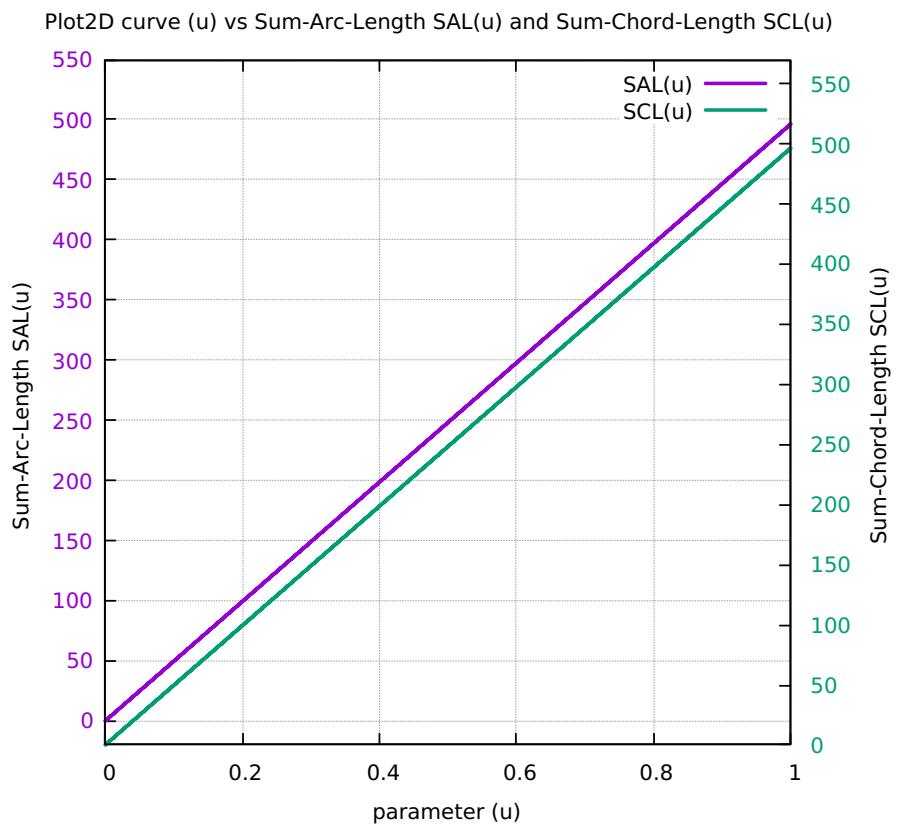


Figure 23: Circle Chord-error in close view 2 scales

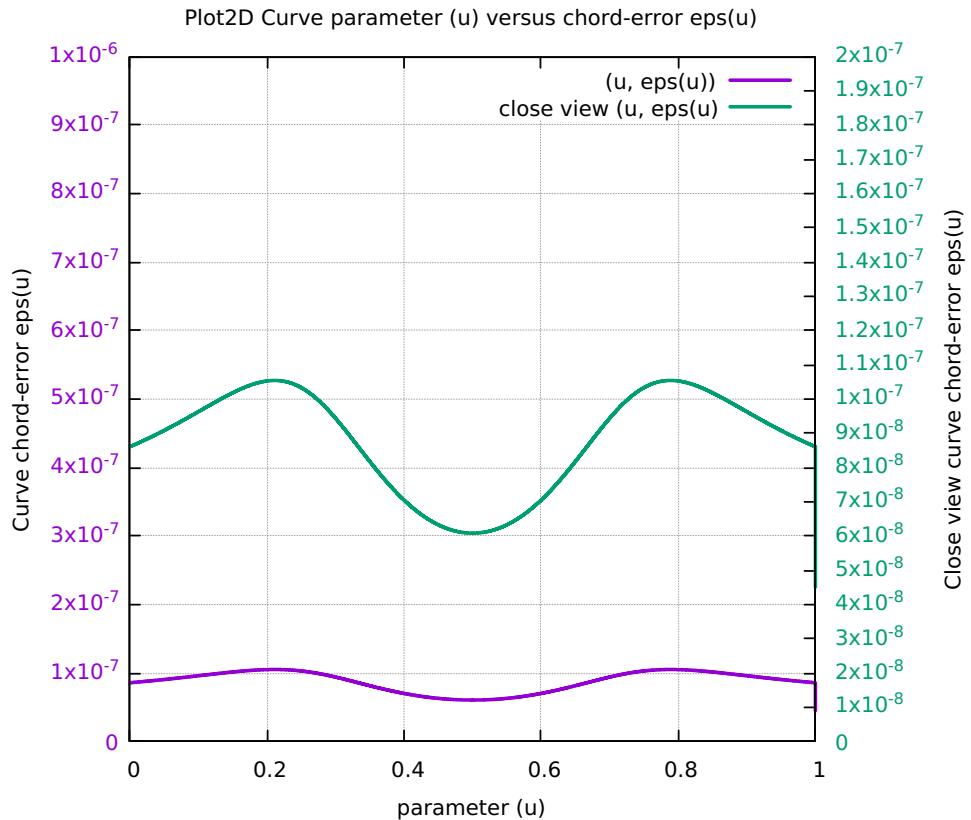


Figure 24: Circle Four Components Feedrate Limit

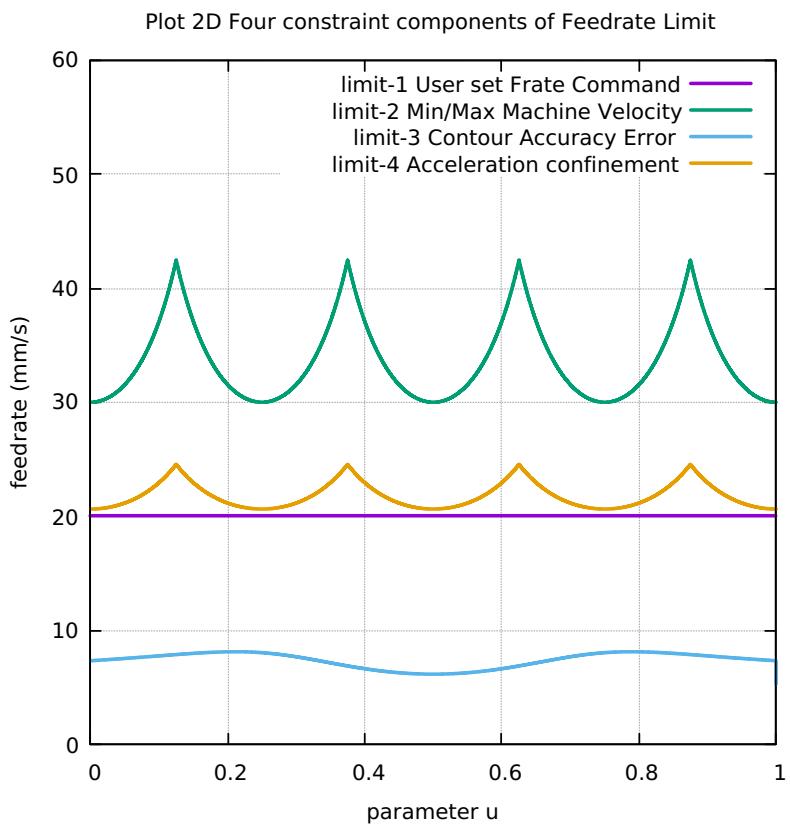


Figure 25: Circle RateCommand RateLimit and Curr-Frate

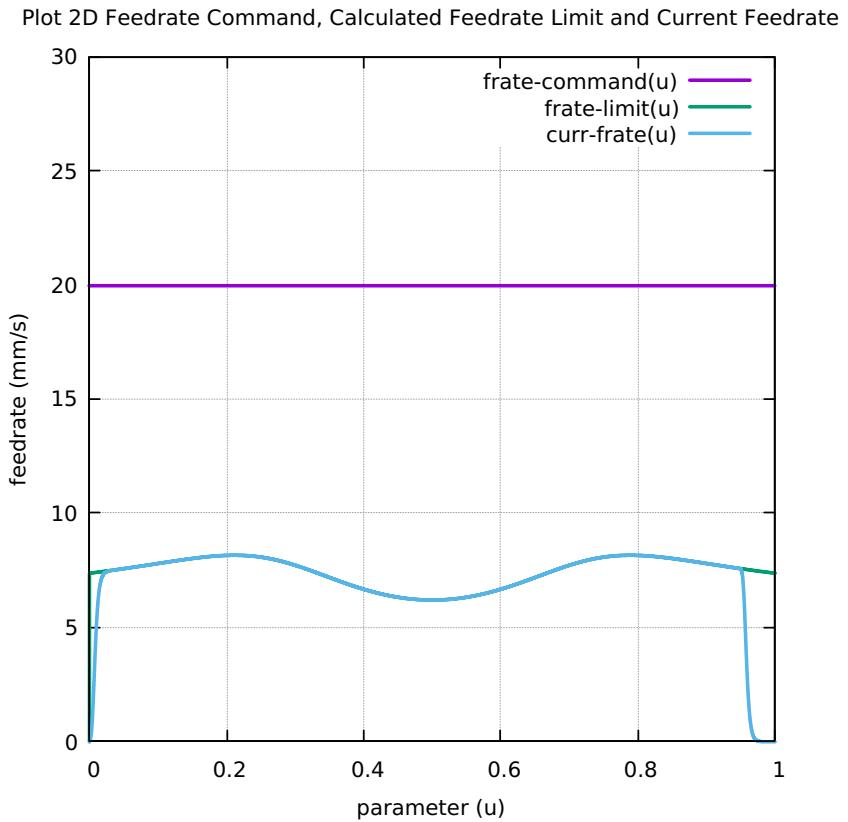


Figure 26: Circle FeedRateLimit minus CurrFeedRate

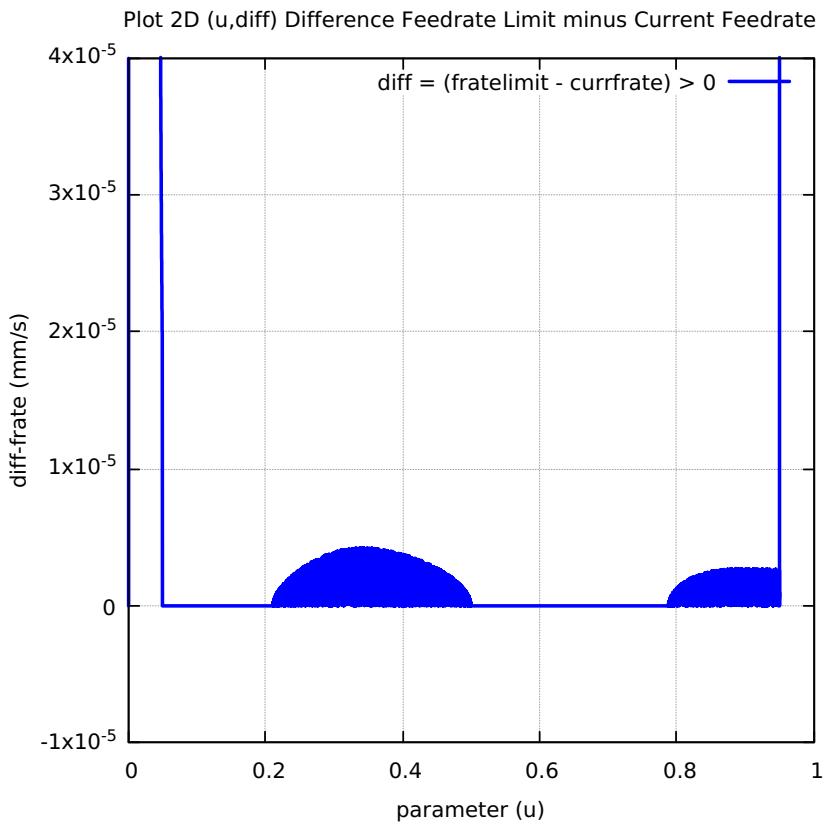


Figure 27: Circle FC20-Nominal X and Y Feedrate Profiles

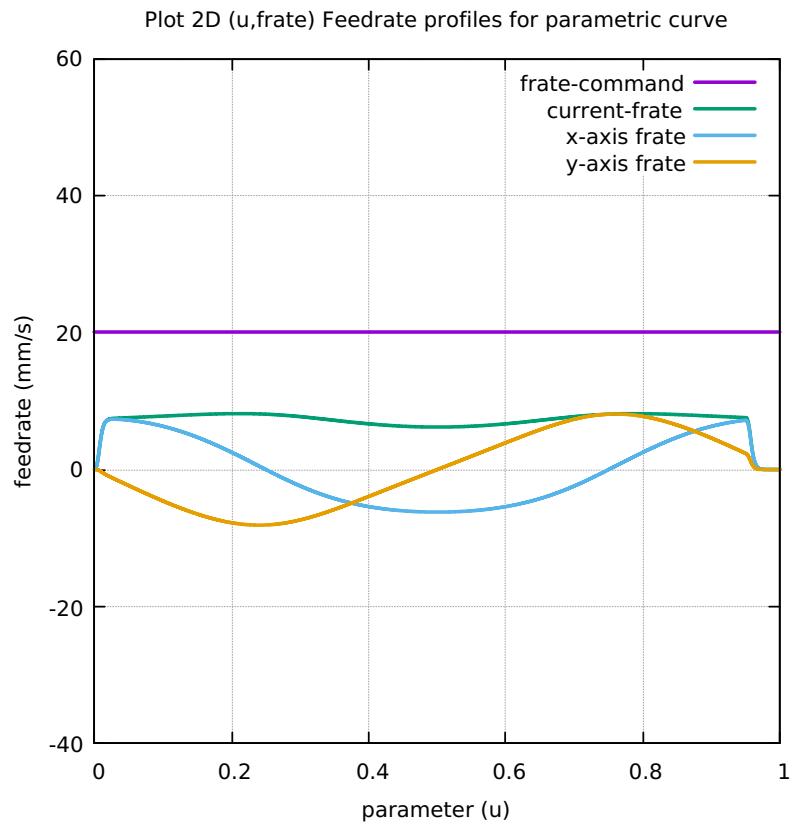


Figure 28: Circle FC20 Nominal Tangential Acceleration

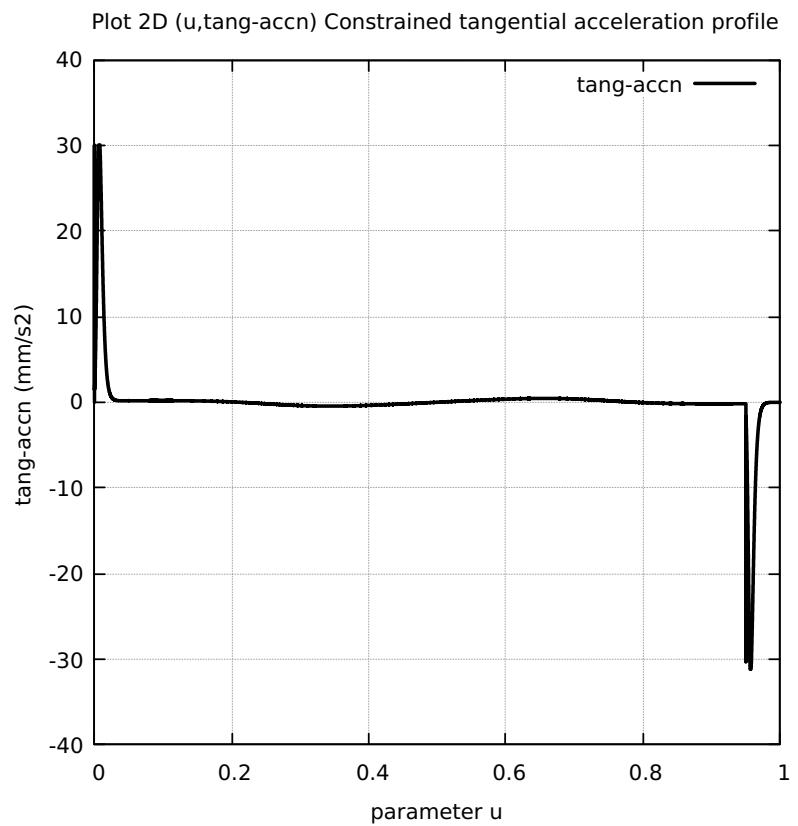


Figure 29: Circle FC20 Nominal Rising S-Curve Profile

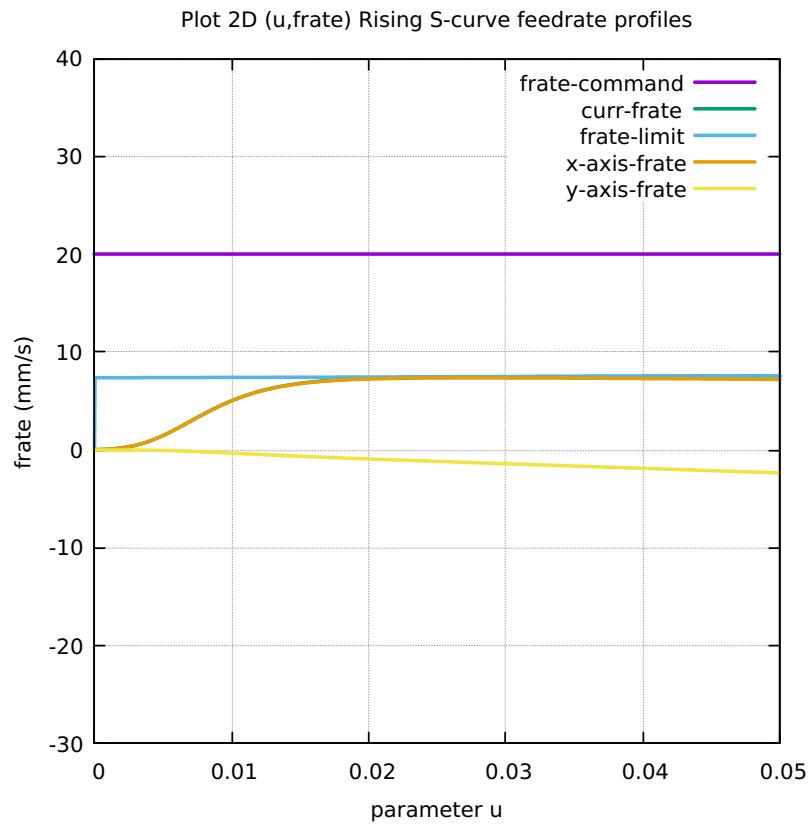


Figure 30: Circle FC20 Nominal Falling S-Curve Profile

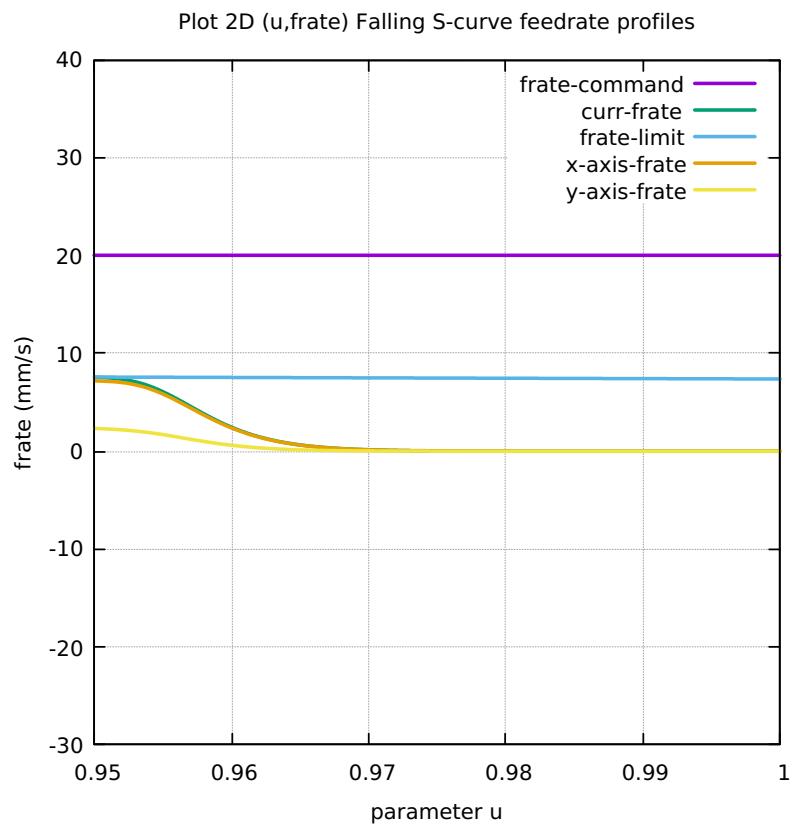


Figure 31: Circle FC10 Colored Feedrate Profile data ngcode

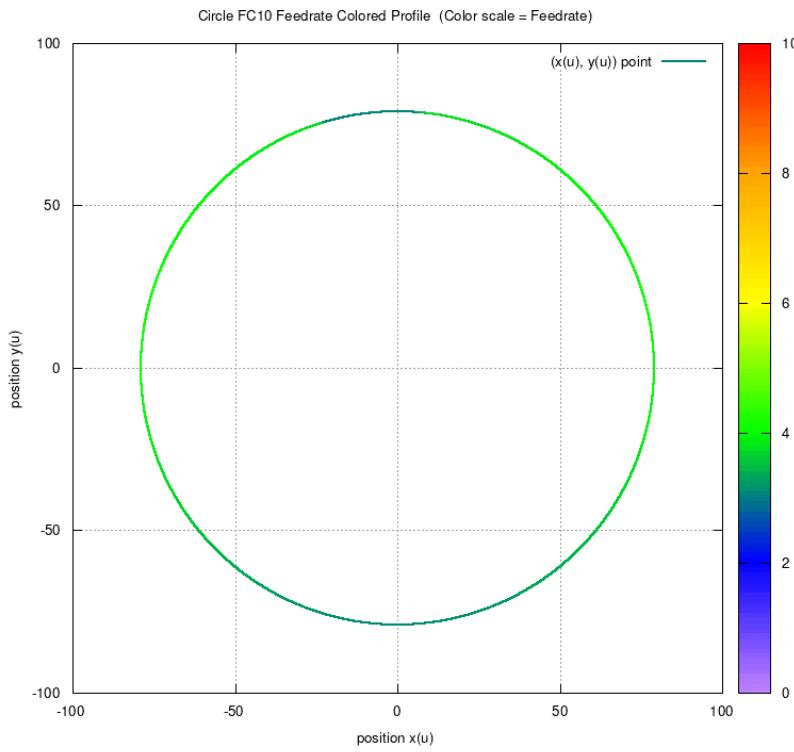


Figure 32: Circle FC20 Colored Feedrate Profile data ngcode

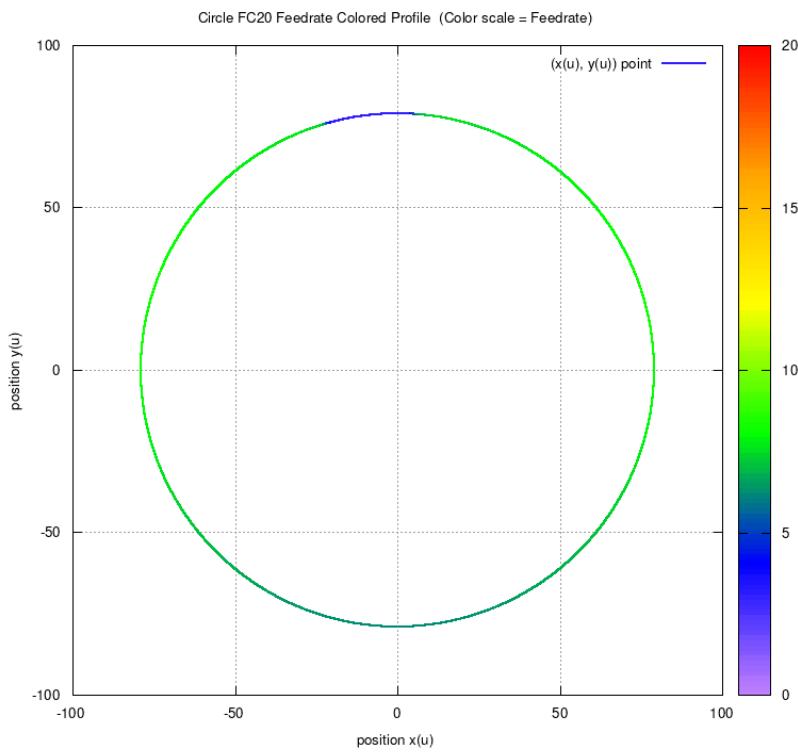


Figure 33: Circle FC30 Colored Feedrate Profile data ngcode

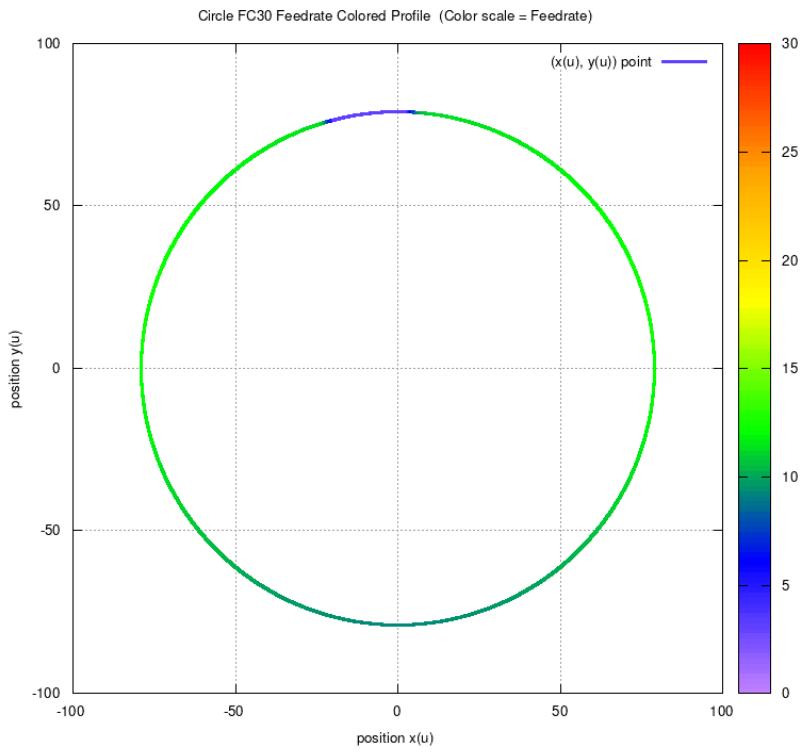


Figure 34: Circle FC40 Colored Feedrate Profile data ngcode

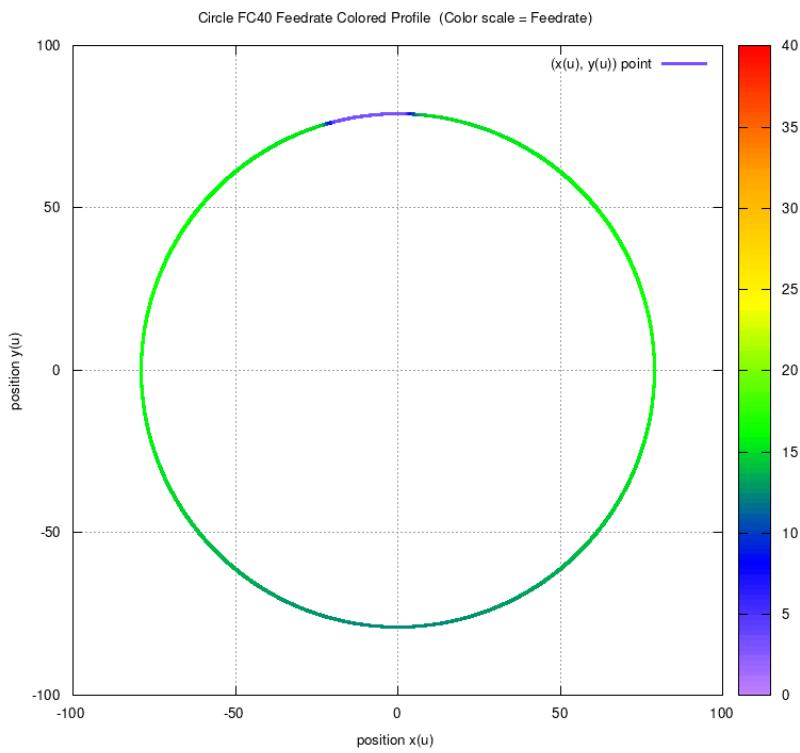


Figure 35: Circle FC10 Tangential Acceleration

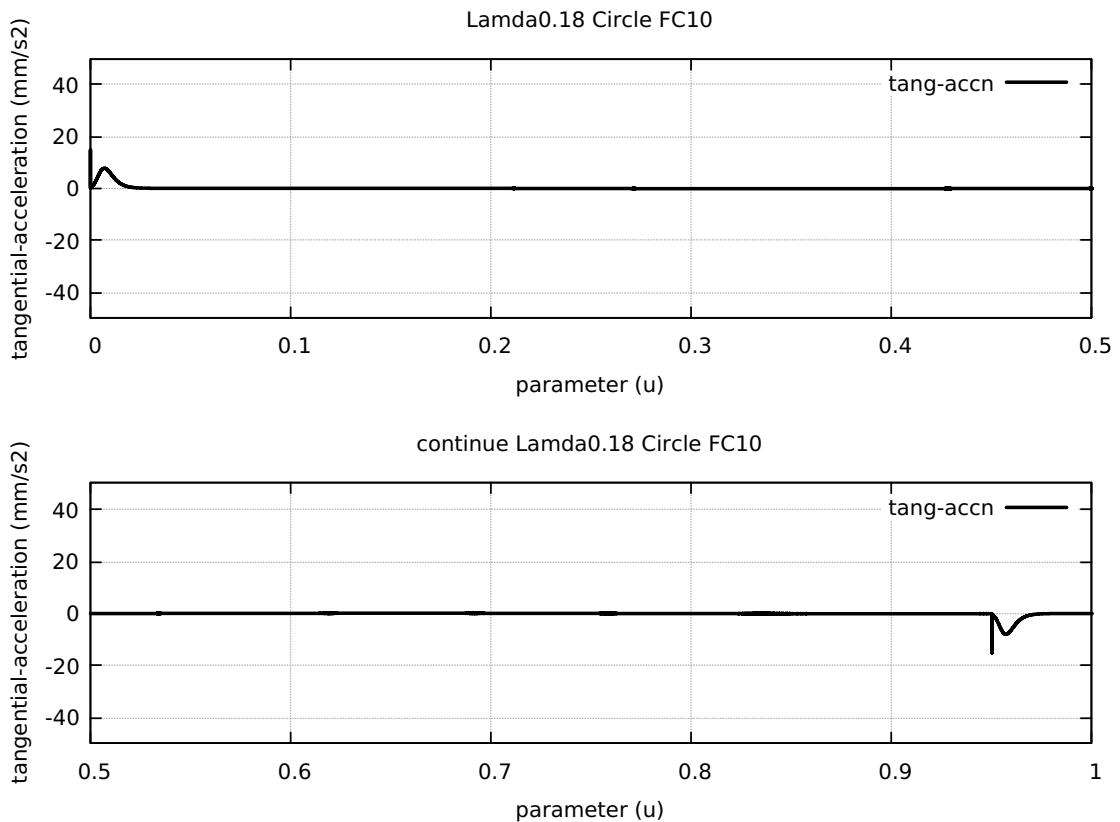


Figure 36: Circle FC20 Tangential Acceleration

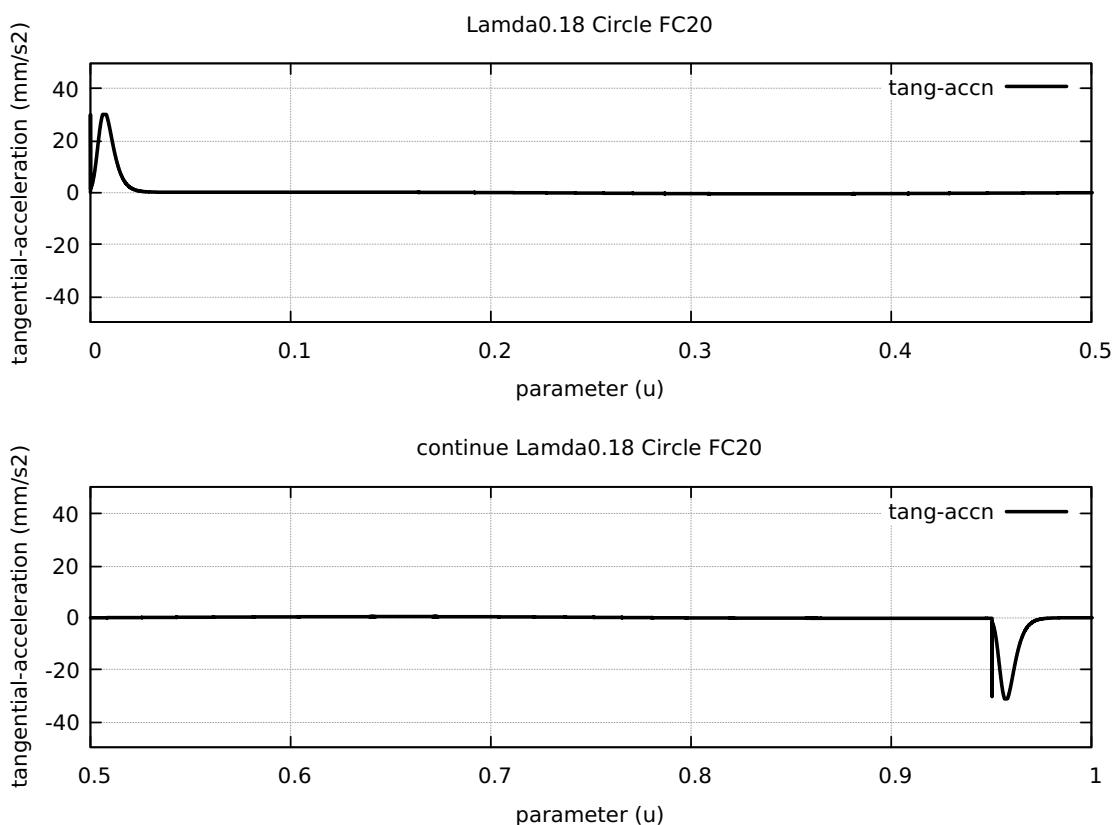


Figure 37: Circle FC30 Tangential Acceleration

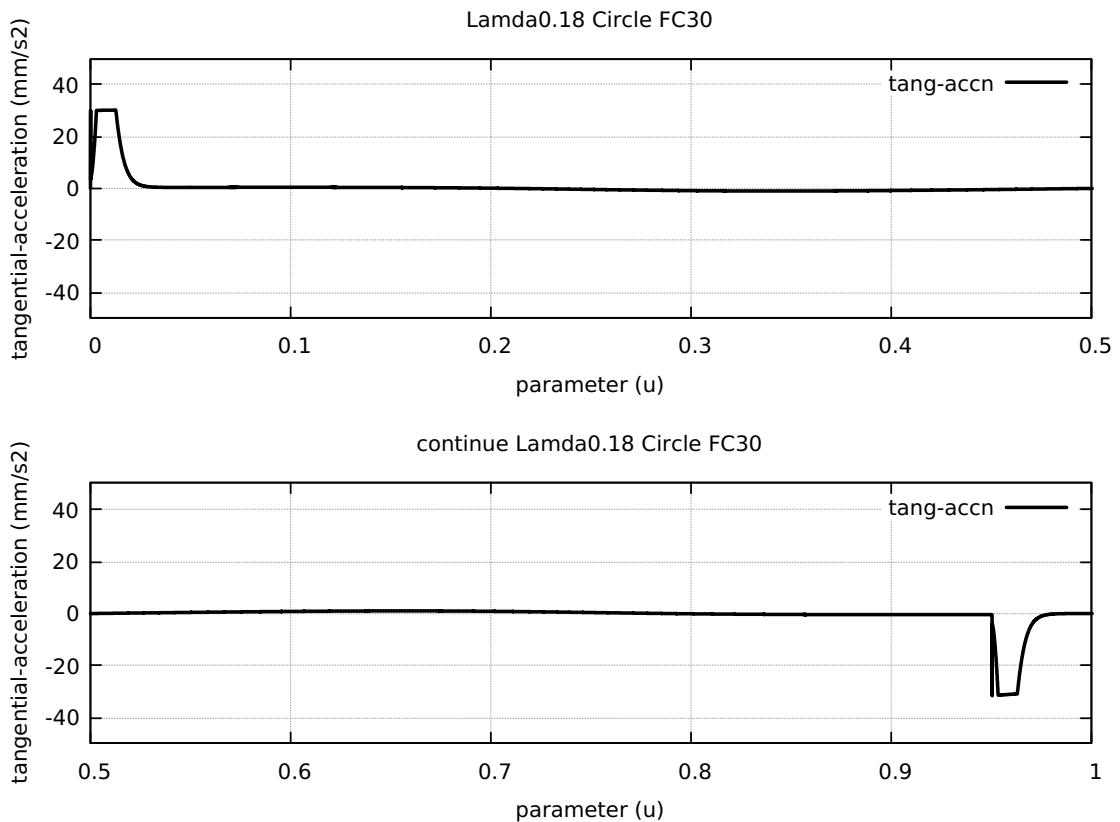


Figure 38: Circle FC40 Tangential Acceleration

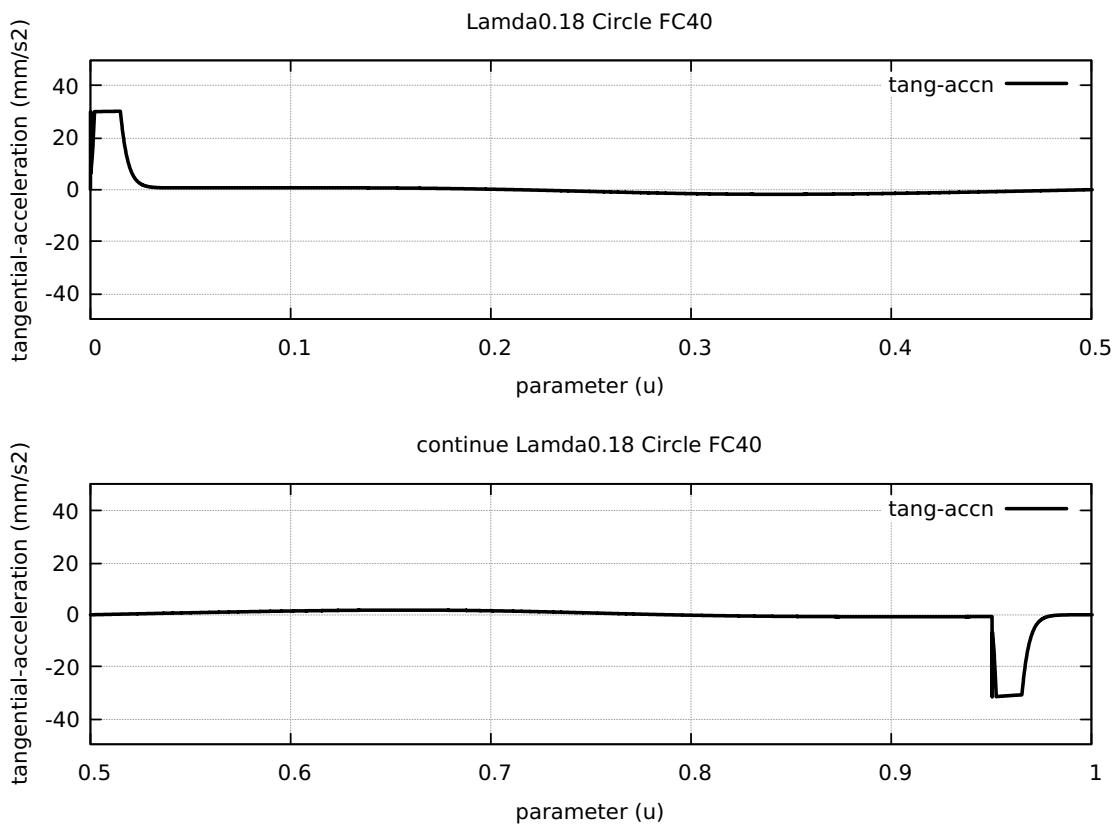


Figure 39: Circle FC20 Nominal Separation NAL and NCL

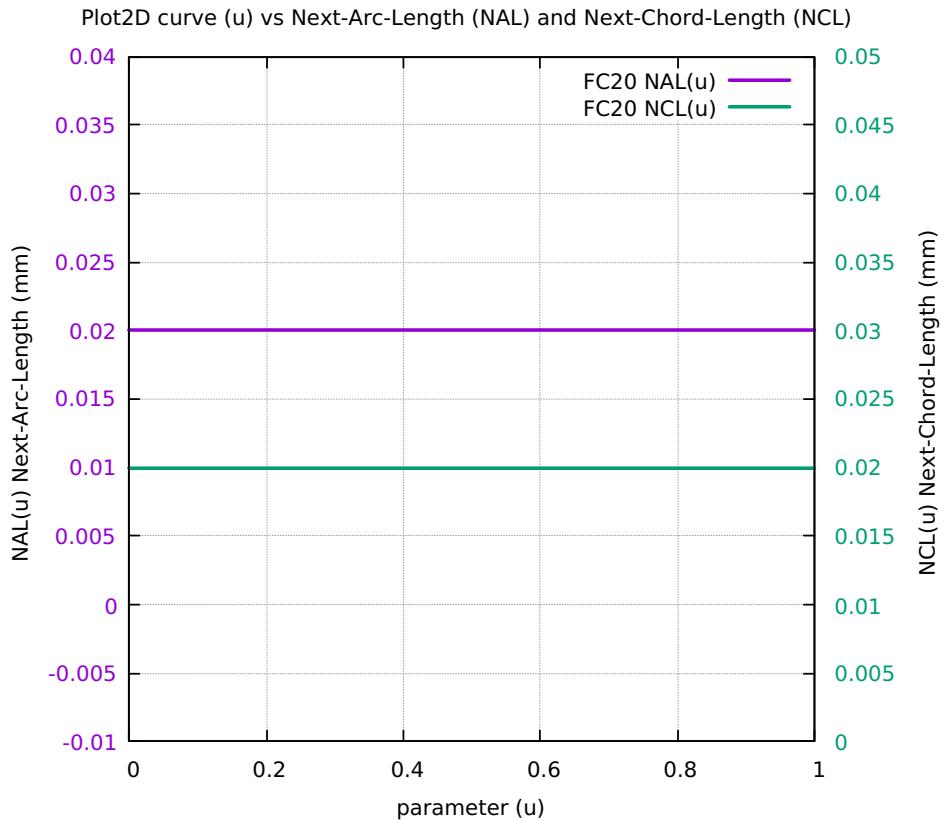


Figure 40: Circle Difference SAL minus SCL for FC10 FC20 FC30 FC40

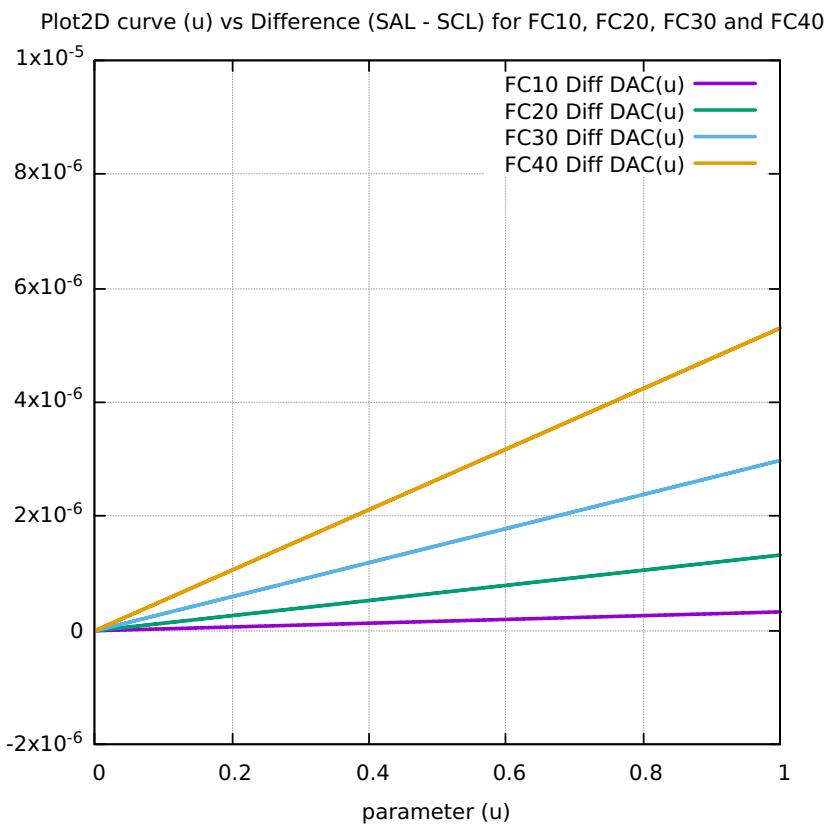


Figure 41: Circle FC10 FRateCmd CurrFRate X-FRate Y-FRate

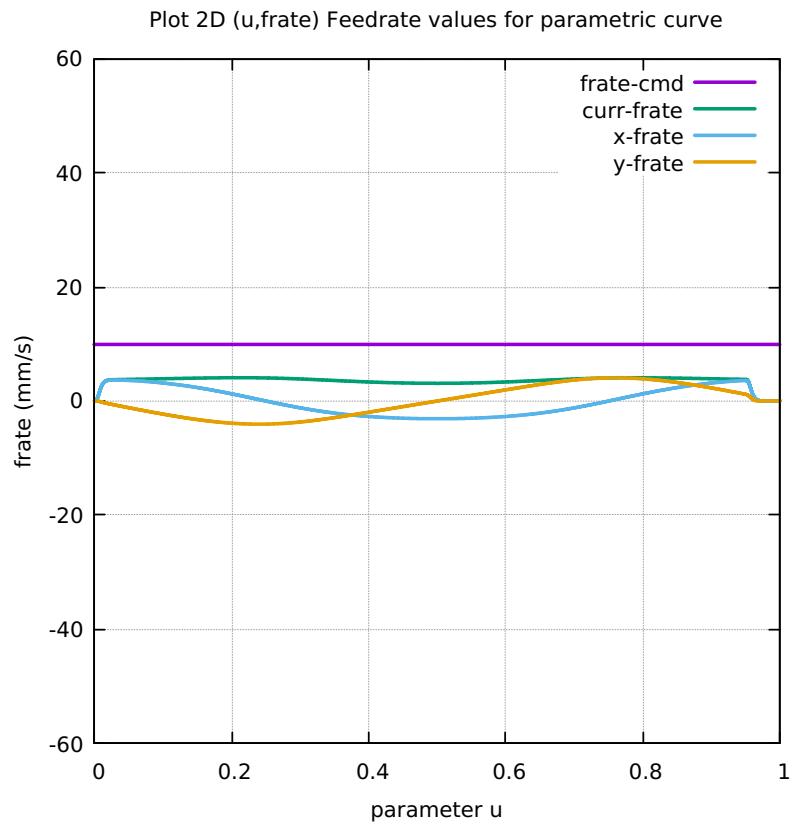


Figure 42: Circle FC20 FRateCmd CurrFRate X-FRate Y-FRate

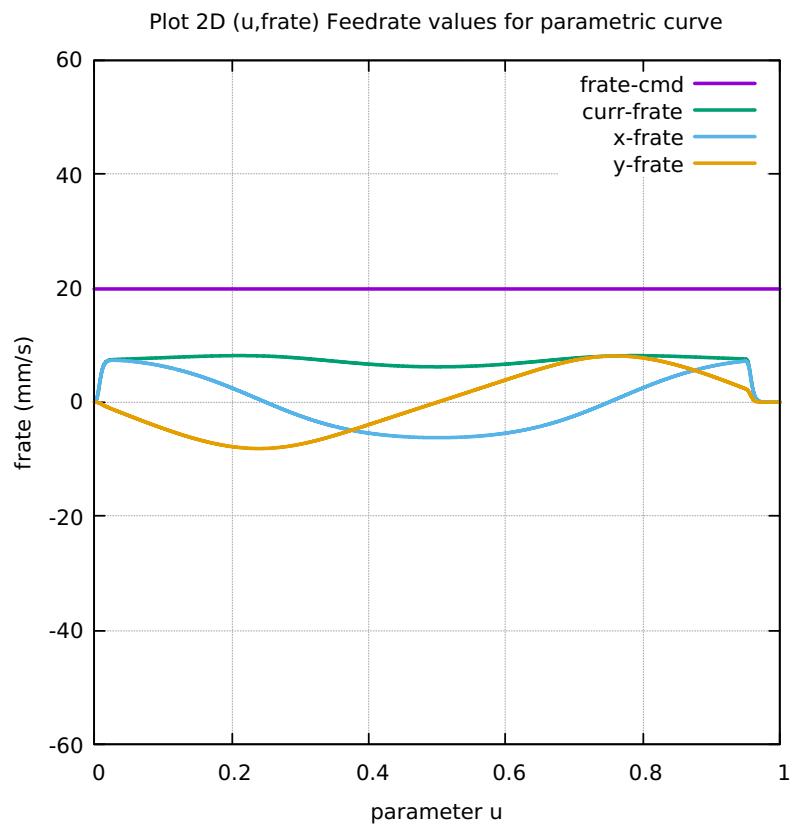


Figure 43: Circle FC30 FRateCmd CurrFRate X-FRate Y-FRate

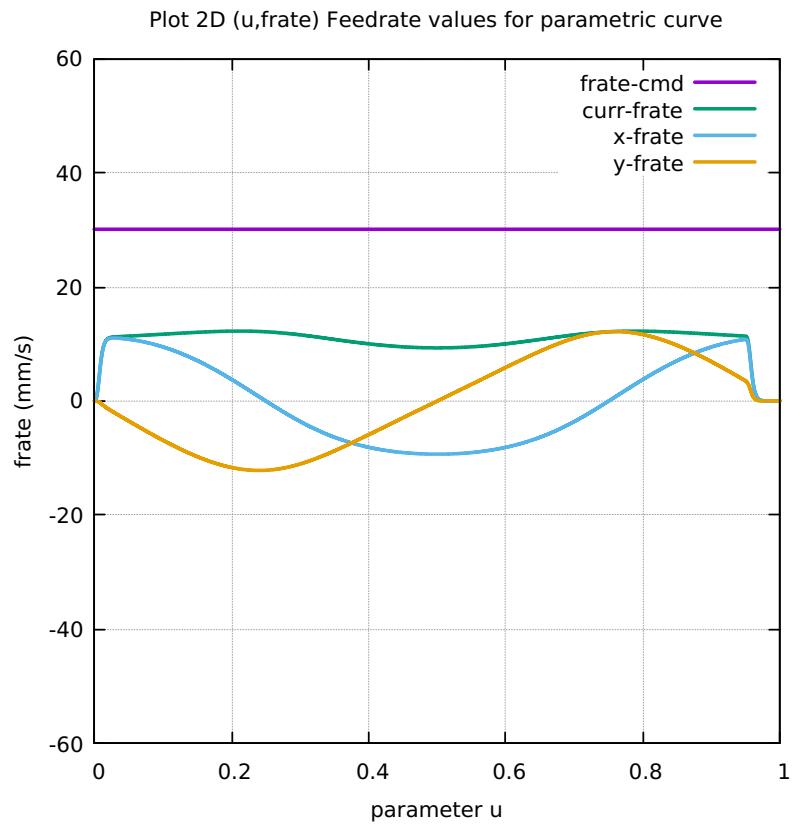


Figure 44: Circle FC40 FRateCmd CurrFRate X-FRate Y-FRate

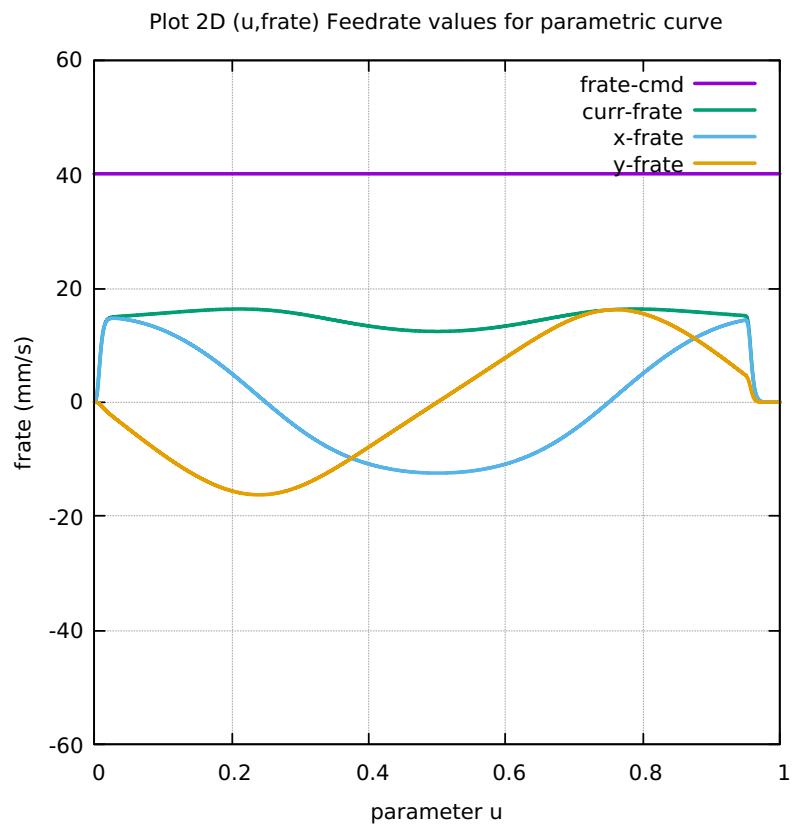


Figure 45: Circle FC10 Four Components FeedrateLimit

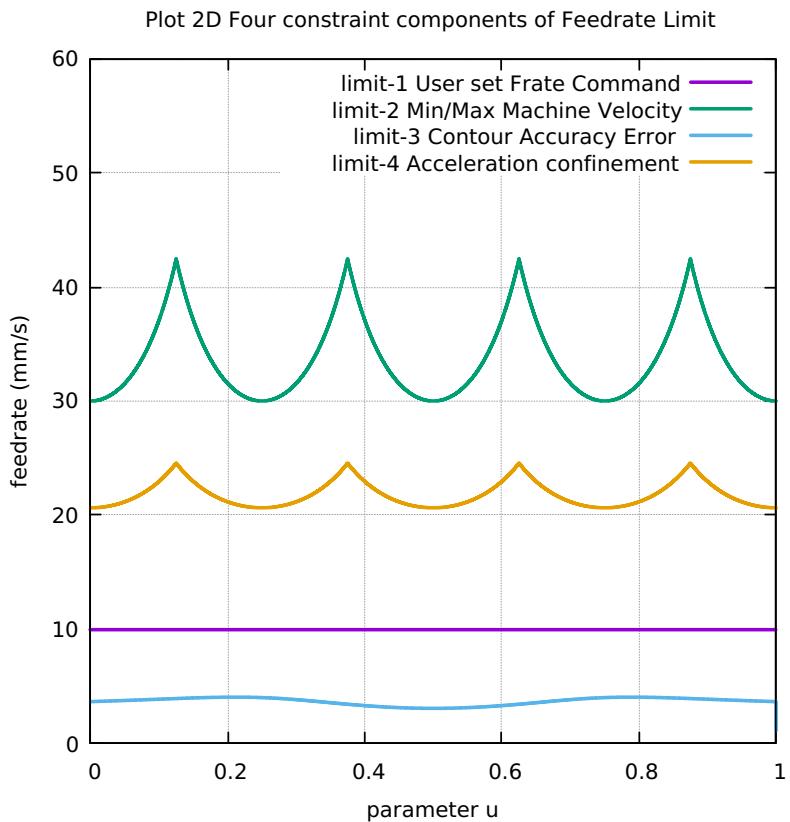


Figure 46: Circle FC20 Four Components FeedrateLimit

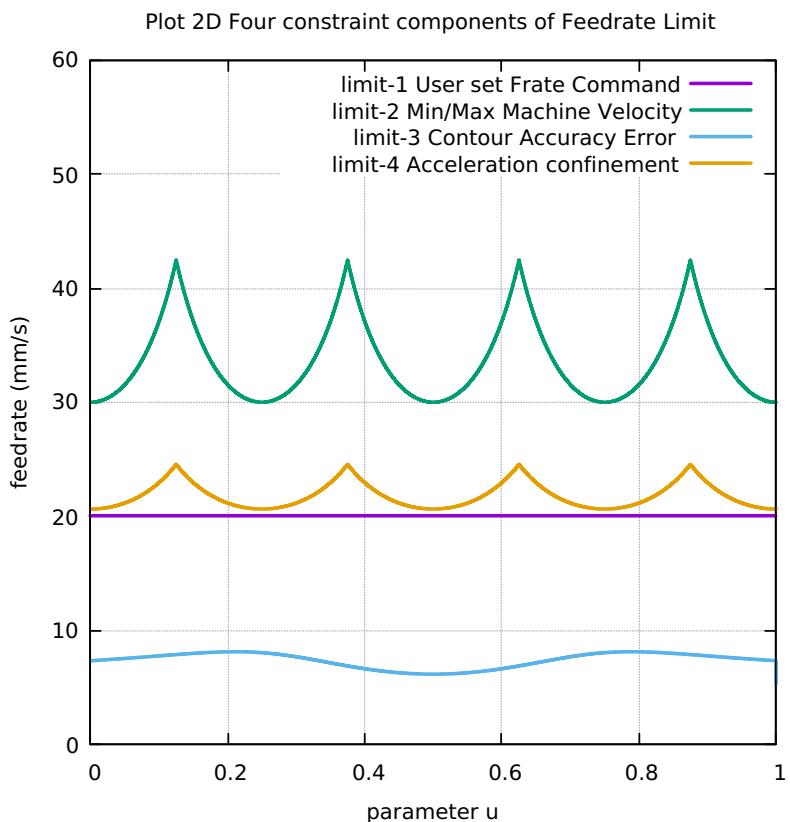


Figure 47: Circle FC30 Four Components FeedrateLimit

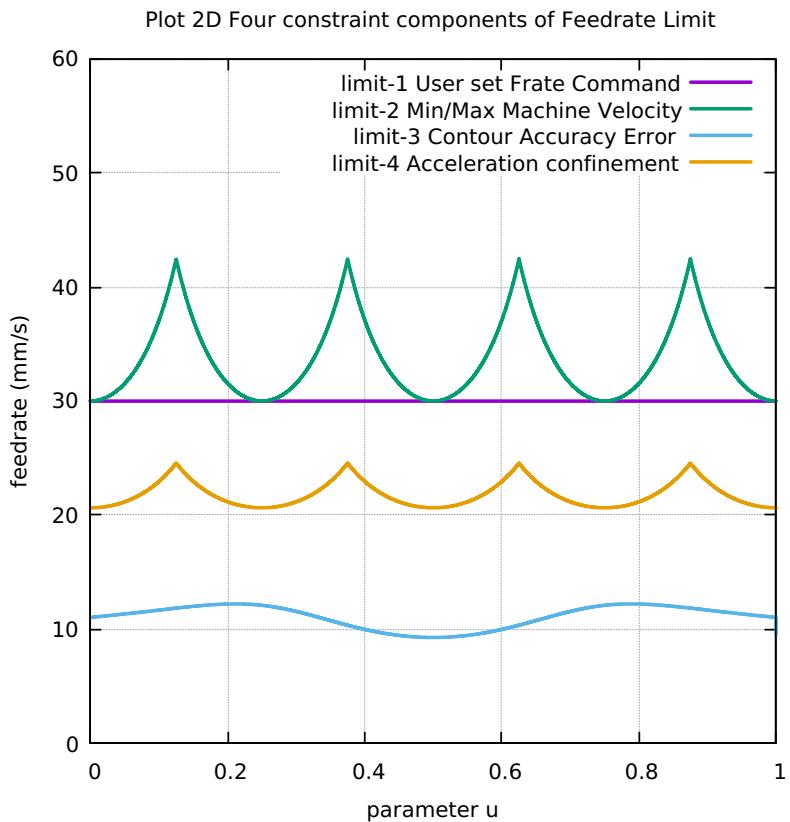


Figure 48: Circle FC40 Four Components FeedrateLimit

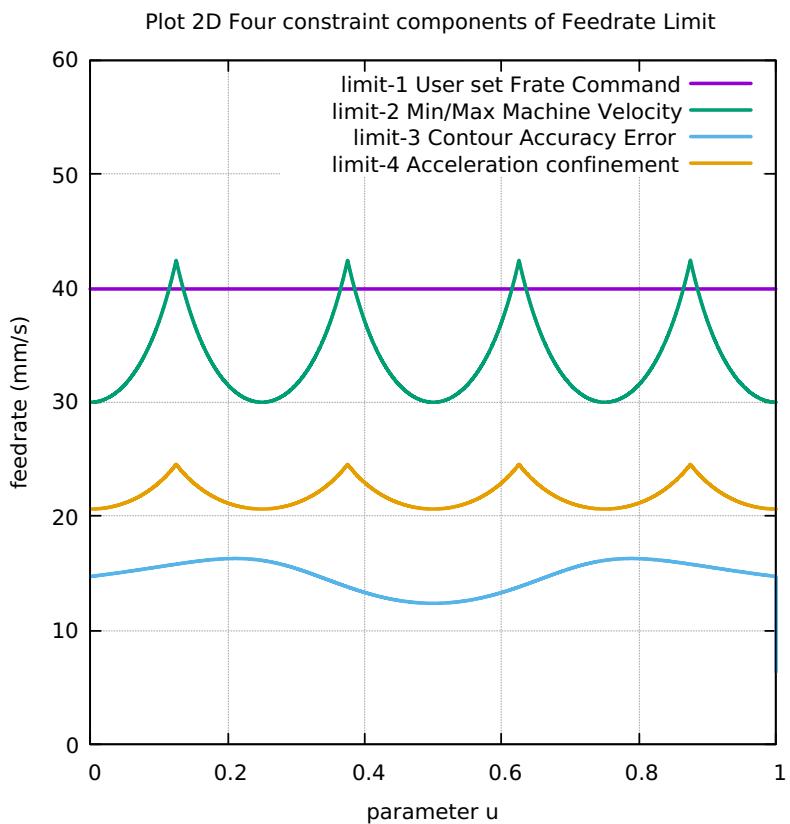


Figure 49: Circle Histogram Points FC10 FC20 FC30 FC40

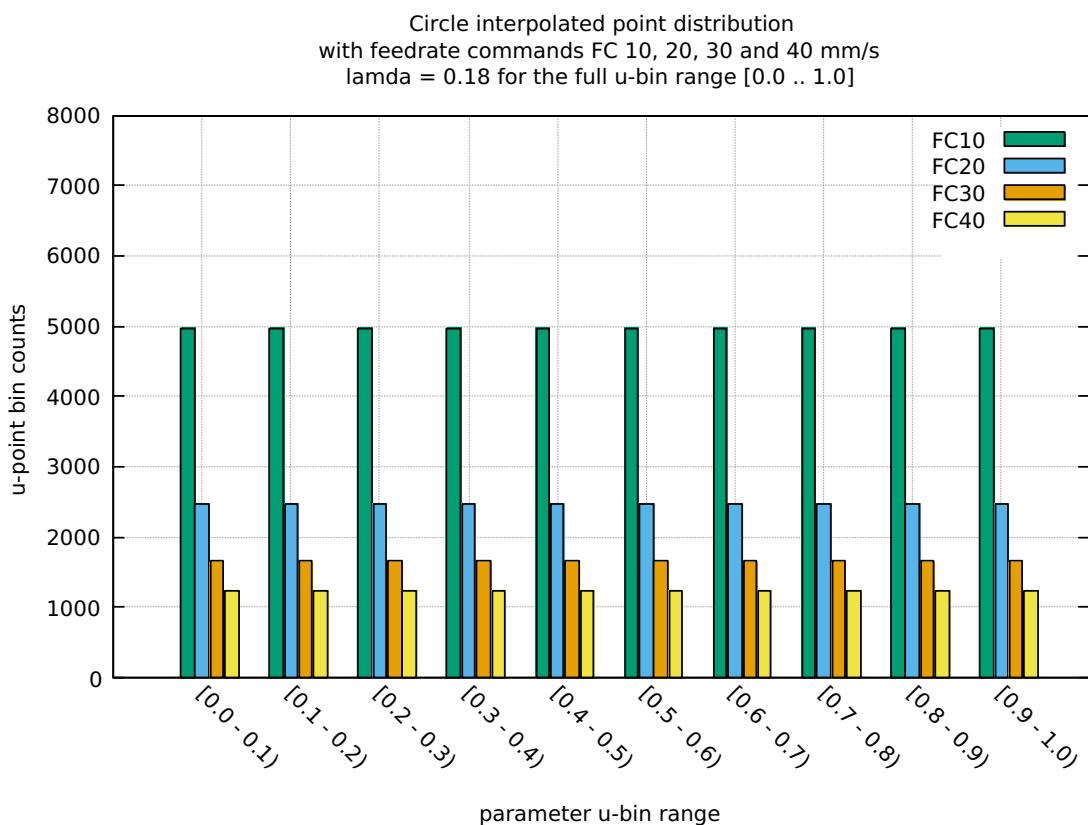


Table 2: Circle Table distribution of interpolated points

BINS	FC10	FC20	FC30	FC40
0.0 - 0.1	4964	2482	1654	1241
0.1 - 0.2	4964	2482	1655	1241
0.2 - 0.3	4964	2482	1655	1241
0.3 - 0.4	4964	2482	1655	1241
0.4 - 0.5	4964	2482	1655	1242
0.5 - 0.6	4964	2483	1655	1241
0.6 - 0.7	4964	2482	1655	1241
0.7 - 0.8	4964	2482	1655	1241
0.8 - 0.9	4964	2482	1654	1242
0.9 - 1.0	4965	2483	1656	1242
Tot Counts	49641	24822	16549	12413

Table 3: Circle Table FC10-20-30-40 Run Performance data

1	Curve Type	CIRCLE		CIRCLE		CIRCLE	
2	User Feedrate Command FC(mm/s)	FC10		FC20		FC30	
3	User Lamda Acceleration Safety Factor	0.18		0.18		0.18	
4	Total Interpolated Points (TIP)	49641		24822		16549	
5	Total Sum-Chord-Error (SCE) (mm)	1.093913738449E-03		2.187639876000E-03		3.2811782287065E-03	
6	Ratio 1 = (SCE/TIP) = Chord-Error/Point	2.203694074232E-08		8.813665347893E-08		1.982824683989E-07	
7	Total Sum-Arc-Length (SAL) (mm)	4.963785816452E+02		4.963771594934E+02		4.963757335444E+02	
8	Total Sum-Chord-Length (SCL) (mm)	4.963775813150E+02		4.963771581682E+02		4.963757305630E+02	
9	Difference = (SAL - SCL) (mm)	3.302195636934E-07		1.325165840171E-06		2.981455850204E-06	
10	Percentage Difference = (SAL - SCL)/SAL	6.652574786747E-08		2.669675295946E-07		6.006449648363E-07	
11	Ratio 2 = (SCE/SCL) = Chord Error/Chord-Length	2.203789163406E-06		4.4077213023407E-06		6.610271383220E-06	
12	Total Sum-Arc-Theta (SAT) (rad)	6.283146611127E+00		6.283002050952E+00		6.282857458743E+00	
13	Total Sum-Arc-Area (SAA) (mm2)	5.235292684461E-05		2.093803820914E-04		4.710353817982E-04	
14	Ratio 3 = (SAA/SCL) = Arc-Area/Chord-Length	2.203789163406E-06		4.4077213023407E-06		6.610271383220E-06	
15	Average-Chord-Error (ACE) (mm)	2.203694074232E-08		8.813665347893E-08		1.98284683989E-07	
16	Average-Arc-Length (AAL) (mm)	9.999568526293E-03		1.999827402173E-02		2.999611636116E-02	
17	Average-Chord-Length (ACL) (mm)	9.999568519641E-03		1.999827396834E-02		2.999611618099E-02	
18	Average-Arc-Theta (AAT) (rad)	1.265742669445E-04		2.531325108155E-04		3.796747316136E-04	
19	Average-Arc-Area (AAA) (mm2)	1.054652031519E-09		8.435614281914E-09		2.846479222856E-08	
20	Algorithm actual runtime on computer (ART) (s)	17.277667419		8.160163431		5.399371536	
							4.053925000

.4 APPENDIX ELLIPSE CURVE

- .4.1 Plot of Ellipse curve [50]**
- .4.2 Ellipse Radius of Curvature [51]**
- .4.3 Ellipse Validation in LinuxCNC [52]**
- .4.4 Ellipse Direction of Travel 3D [53]**
- .4.5 Ellipse First and Second Order Taylor's Approx [54]**
- .4.6 Ellipse First minus Second Order Taylor's Approx [55]**
- .4.7 Ellipse Separate First and Second Order Taylor's Approx [56]**
- .4.8 Ellipse Separation SAL and SCL [57]**
- .4.9 Ellipse Chord-error in close view 2 scales [58]**
- .4.10 Ellipse Four Components Feedrate Limit [59]**
- .4.11 Ellipse FrateCommand FrateLimit and Curr-Frate [60]**
- .4.12 Ellipse FeedRateLimit minus CurrFeedRate [61]**
- .4.13 Ellipse FC20-Nominal X and Y Feedrate Profiles [62]**
- .4.14 Ellipse FC20 Nominal Tangential Acceleration [63]**
- .4.15 Ellipse FC20 Nominal Rising S-Curve Profile [64]**
- .4.16 Ellipse FC20 Nominal Falling S-Curve Profile [65]**
- .4.17 Ellipse FC10 Colored Feedrate Profile data ngcode [66]**
- .4.18 Ellipse FC20 Colored Feedrate Profile data ngcode [67]**

- .4.19 Ellipse FC30 Colored Feedrate Profile data ngcode [68]
- .4.20 Ellipse FC40 Colored Feedrate Profile data ngcode [69]
- .4.21 Ellipse FC10 Tangential Acceleration [70]
- .4.22 Ellipse FC20 Tangential Acceleration [71]
- .4.23 Ellipse FC30 Tangential Acceleration [72]
- .4.24 Ellipse FC40 Tangential Acceleration [73]
- .4.25 Ellipse FC20 Nominal Separation NAL and NCL [74]
- .4.26 Ellipse SAL minus SCL for FC10 FC20 FC30 FC40 [75]
- .4.27 Ellipse FC10 FrateCmd CurrFrate X-Frate Y-Frate [76]
- .4.28 Ellipse FC20 FrateCmd CurrFrate X-Frate Y-Frate [77]
- .4.29 Ellipse FC30 FrateCmd CurrFrate X-Frate Y-Frate [78]
- .4.30 Ellipse FC40 FrateCmd CurrFrate X-Frate Y-Frate [79]
- .4.31 Ellipse FC10 Four Components FeedrateLimit [80]
- .4.32 Ellipse FC20 Four Components FeedrateLimit [81]
- .4.33 Ellipse FC30 Four Components FeedrateLimit [82]
- .4.34 Ellipse FC40 Four Components FeedrateLimit [83]
- .4.35 Ellipse Histogram Points FC10 FC20 FC30 FC40 [84]
- .4.36 Ellipse Table distribution of interpolated points [4]
- .4.37 Ellipse Table FC10-20-30-40 Run Performance data [5]

Figure 50: Plot of Ellipse curve

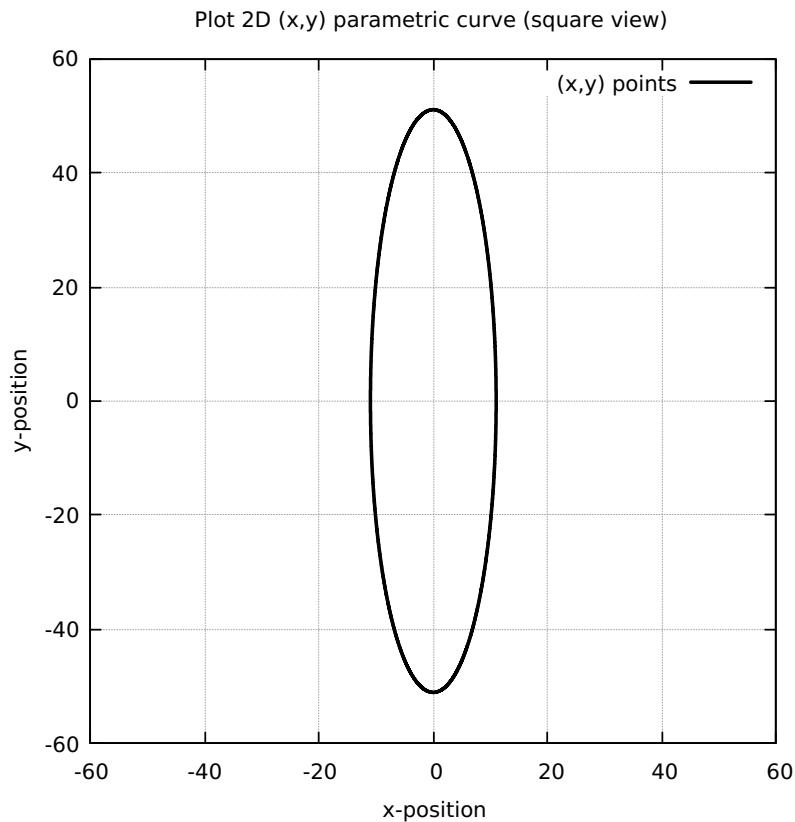


Figure 51: Ellipse Radius of Curvature

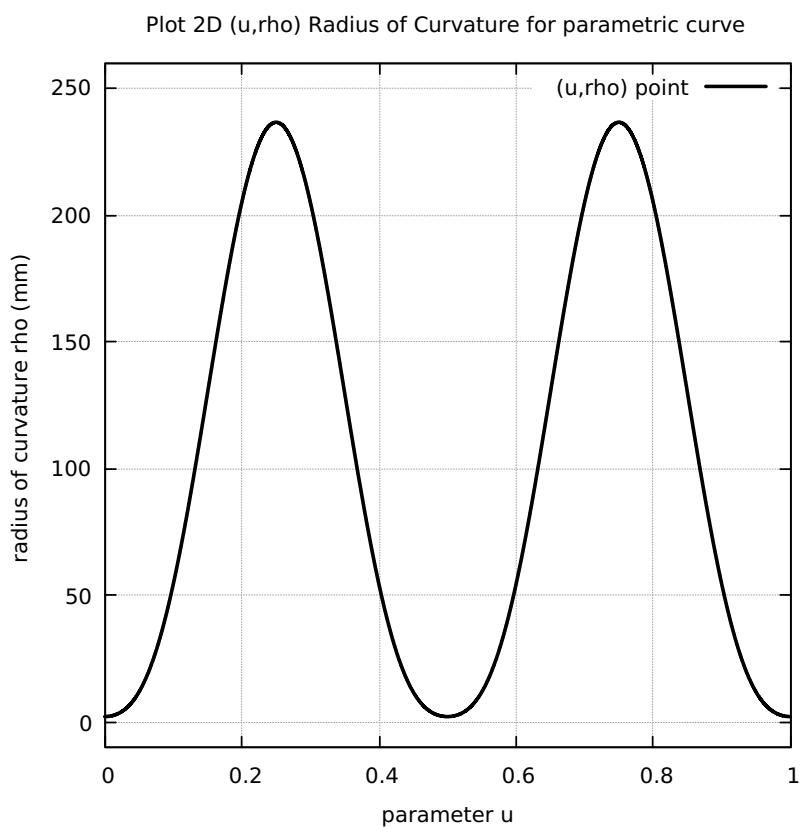


Figure 52: Ellipse Validation in LinuxCNC

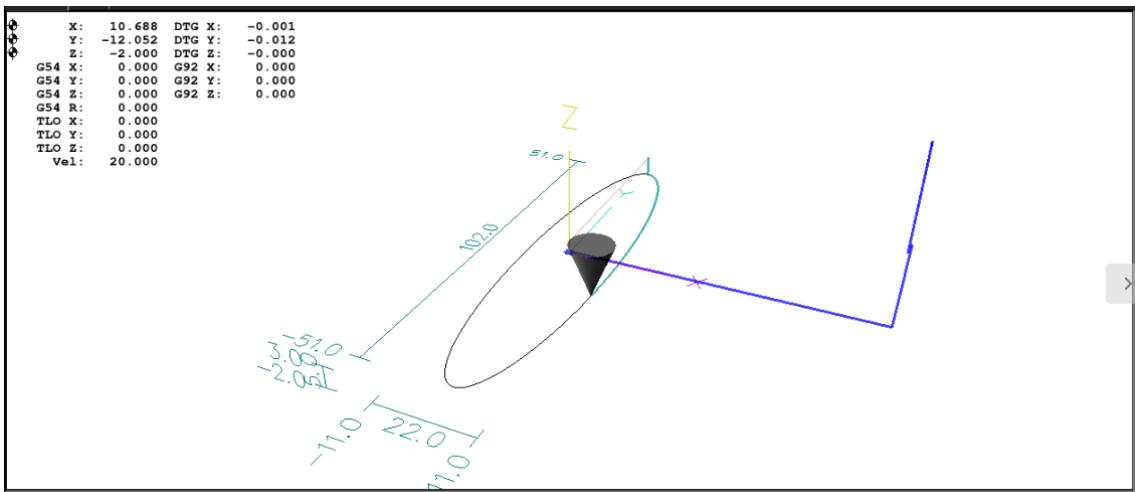


Figure 53: Ellipse Direction of Travel 3D

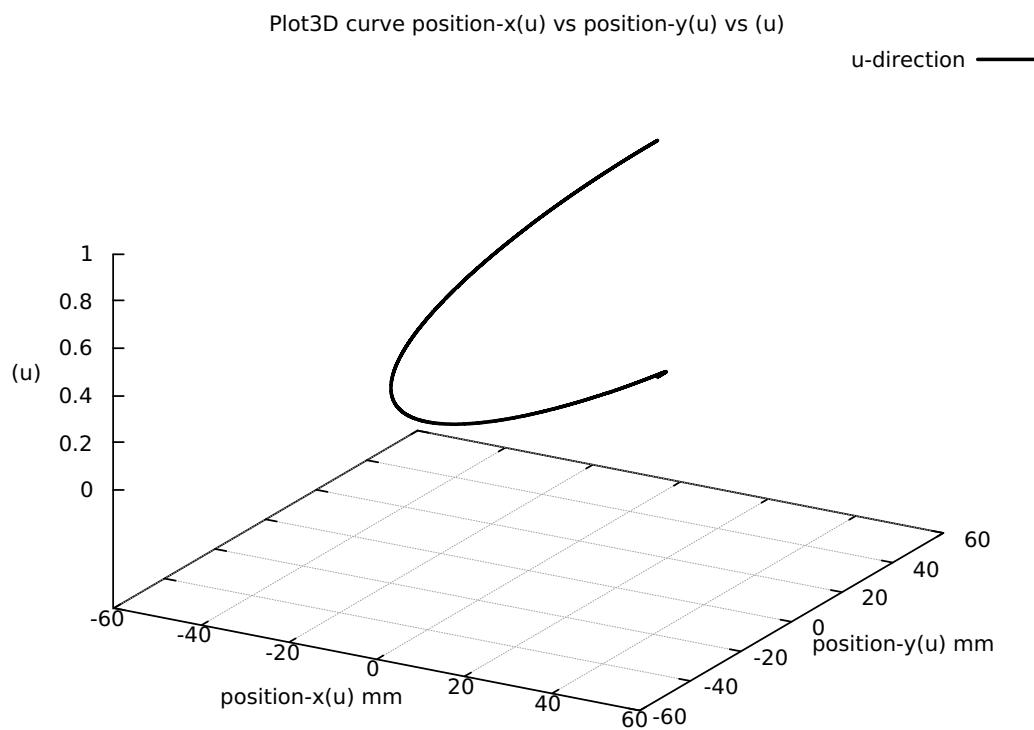


Figure 54: Ellipse First and Second Order Taylor's Approximation

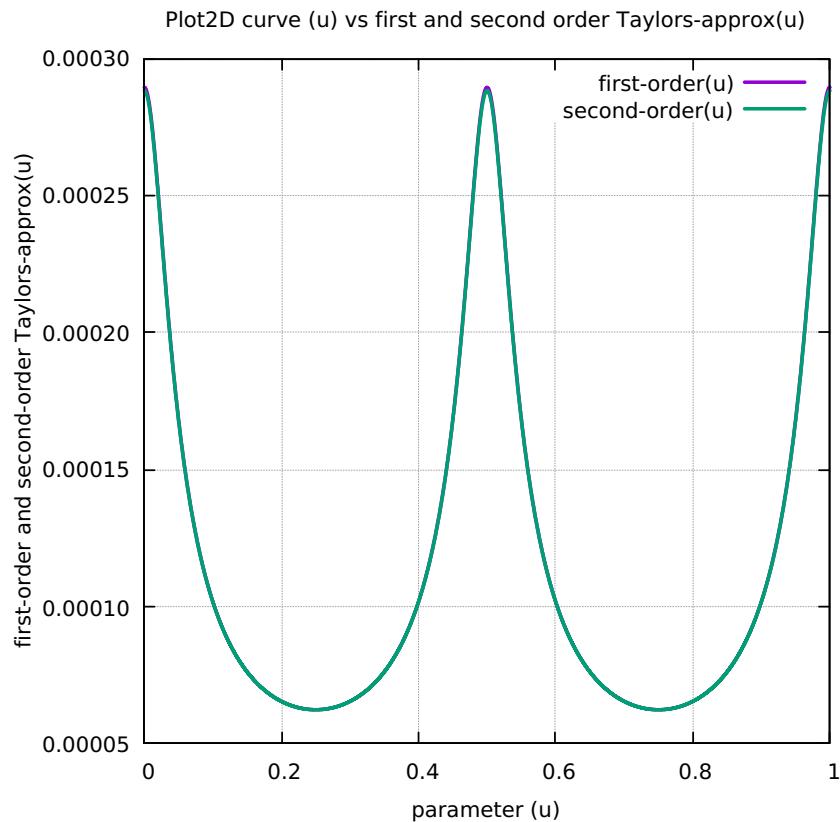


Figure 55: Ellipse First minus Second Order Taylor's Approximation

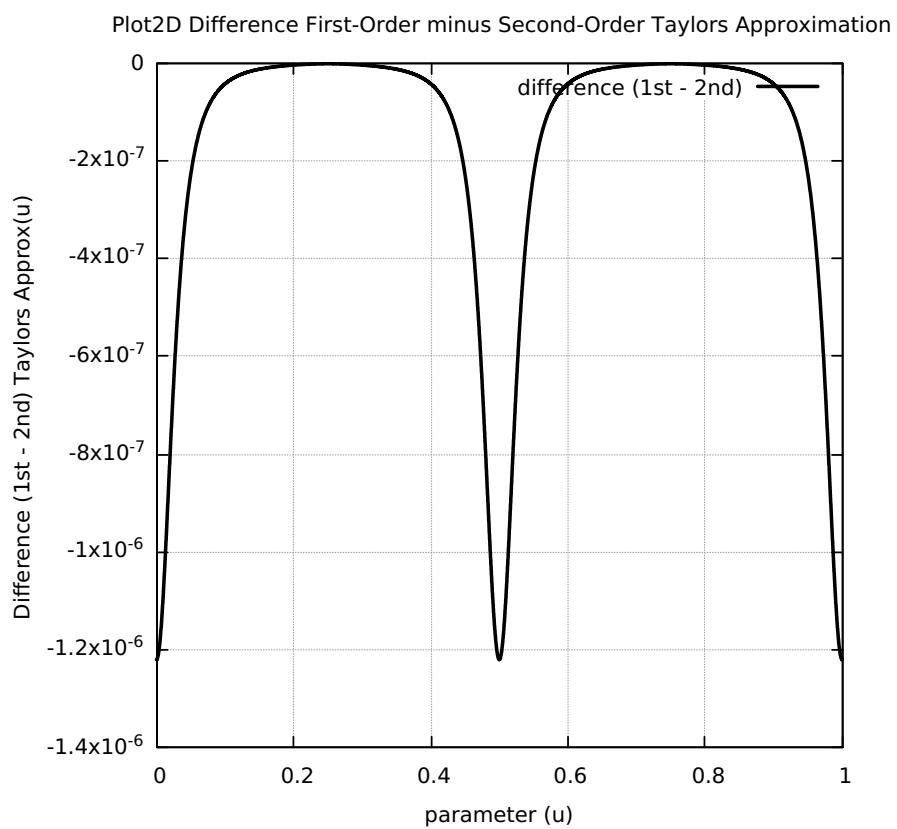


Figure 56: Ellipse Separation First and Second Order Taylor's Approximation

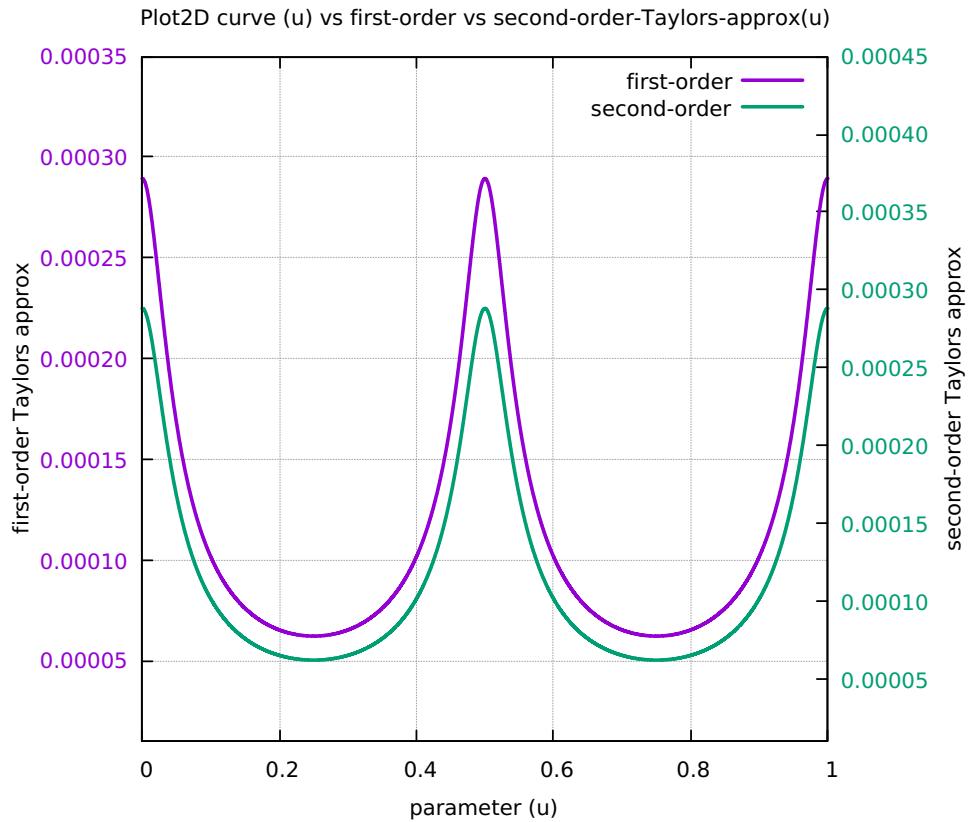


Figure 57: Ellipse Separation SAL and SCL

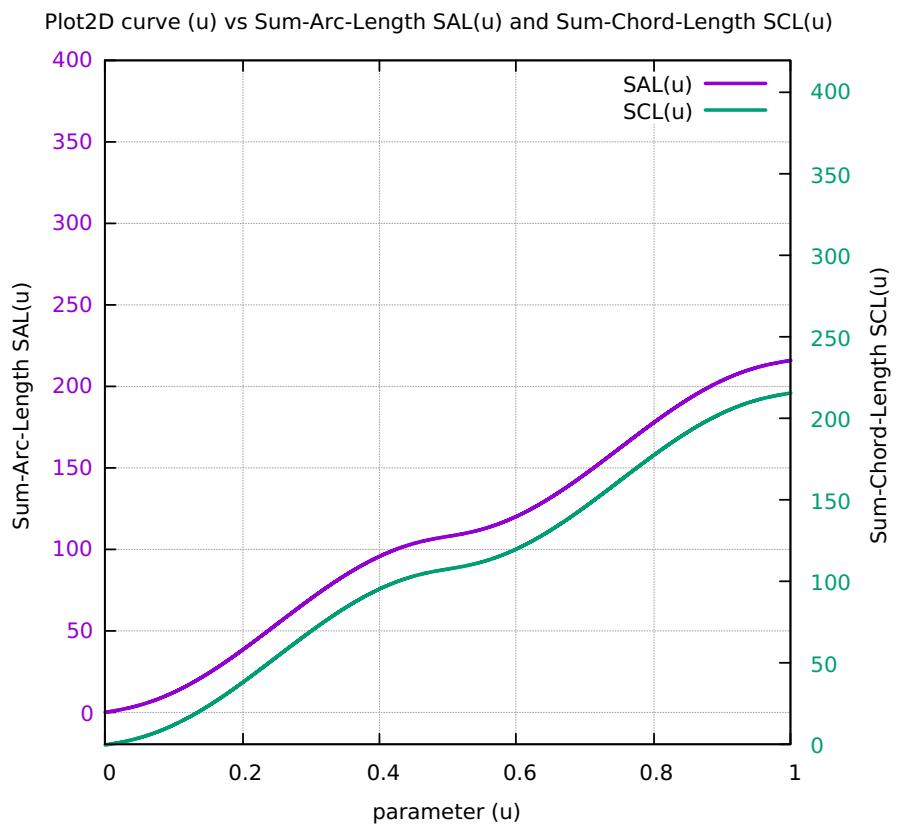


Figure 58: Ellipse Chord-error in close view 2 scales

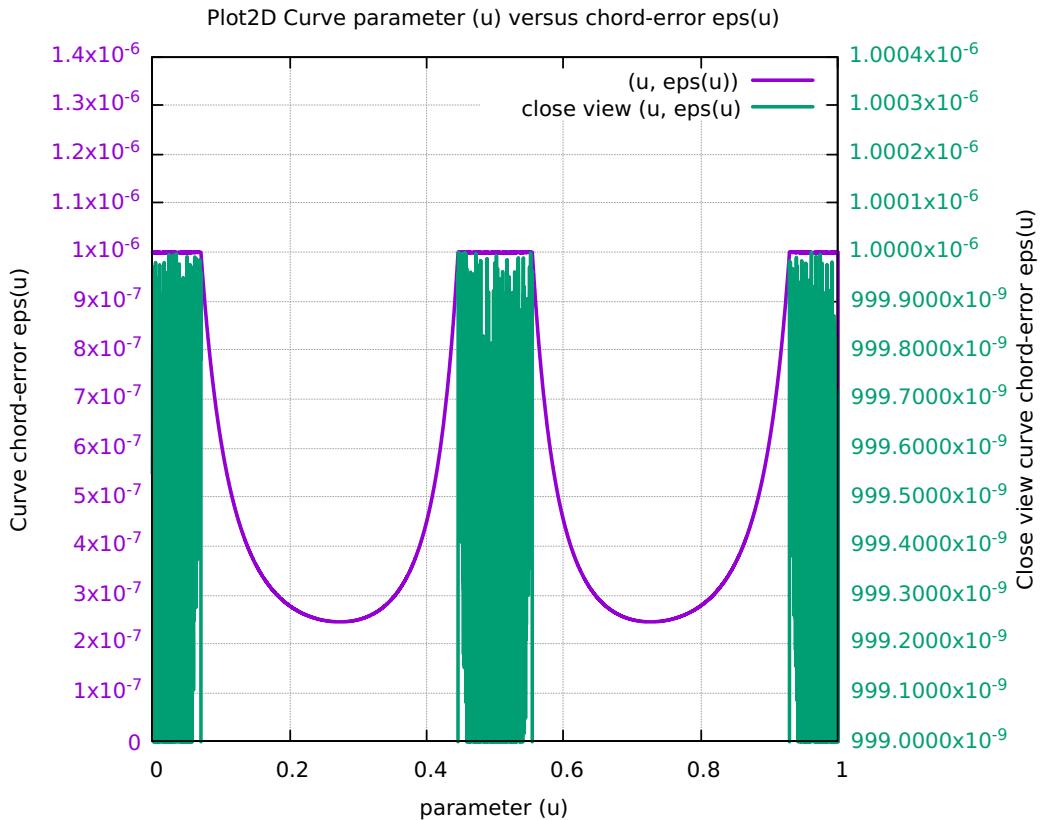


Figure 59: Ellipse Four Components Feedrate Limit

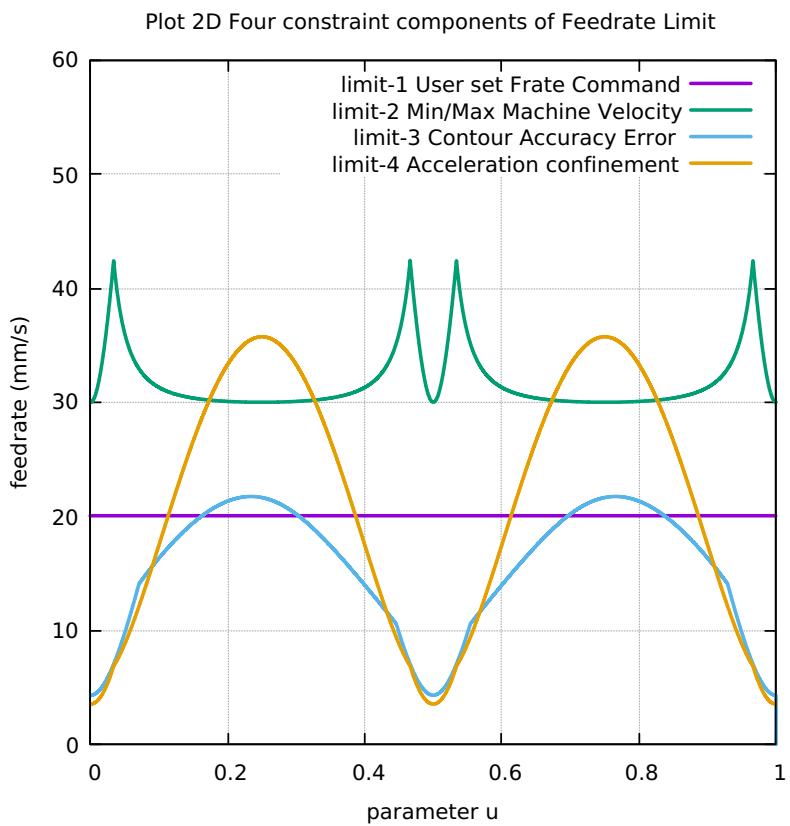


Figure 60: Ellipse FrateCommand FrateLimit and Curr-Frate

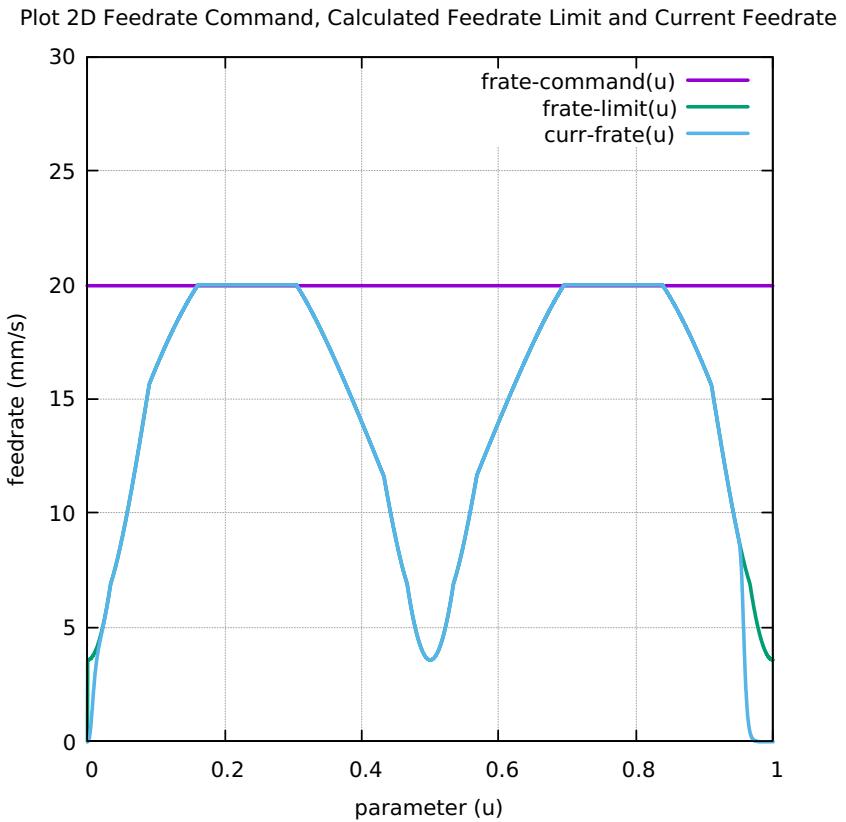


Figure 61: Ellipse FeedRateLimit minus CurrFeedRate

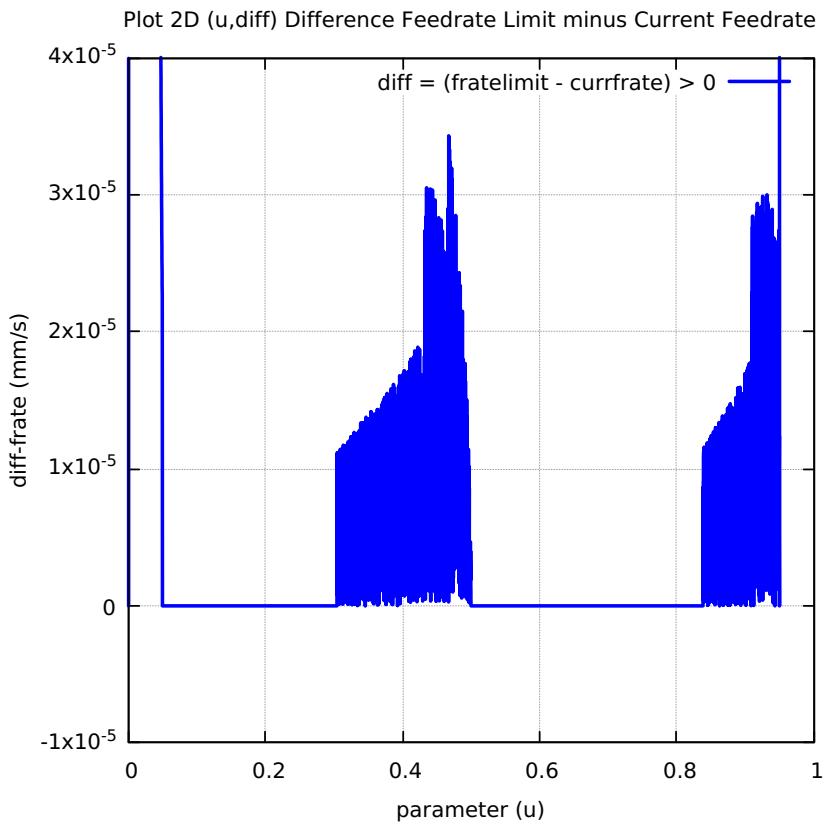


Figure 62: Ellipse FC20-Nominal X and Y Feedrate Profiles

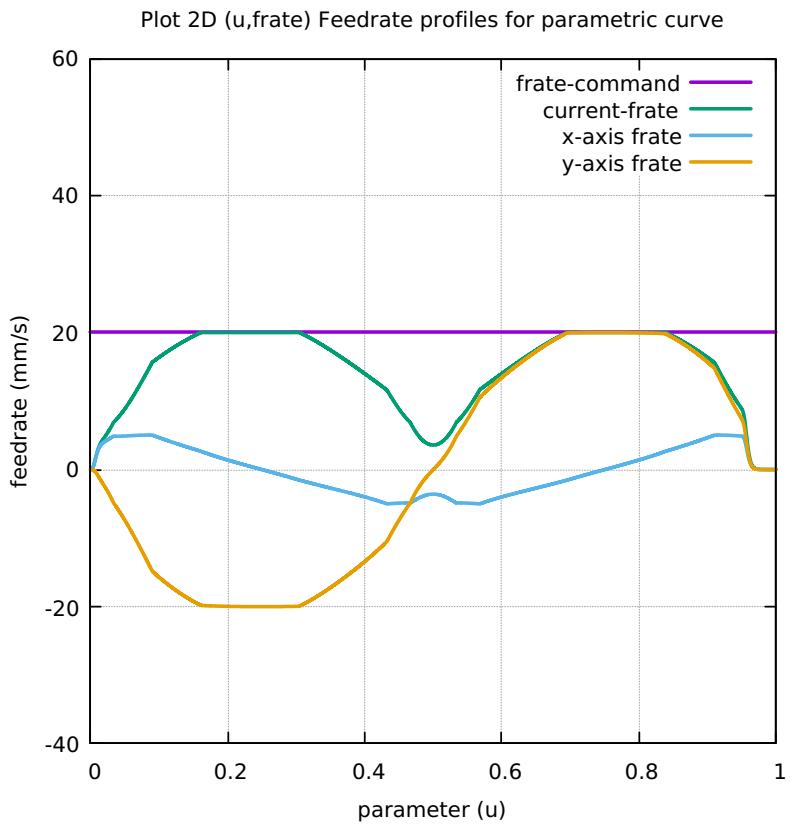


Figure 63: Ellipse FC20 Nominal Tangential Acceleration

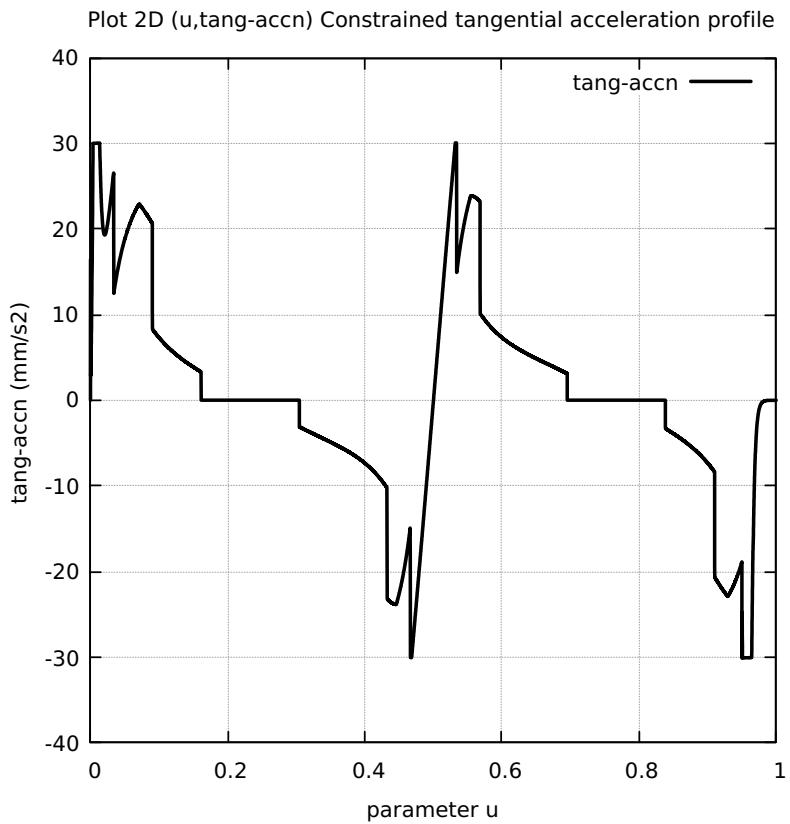


Figure 64: Ellipse FC20 Nominal Rising S-Curve Profile

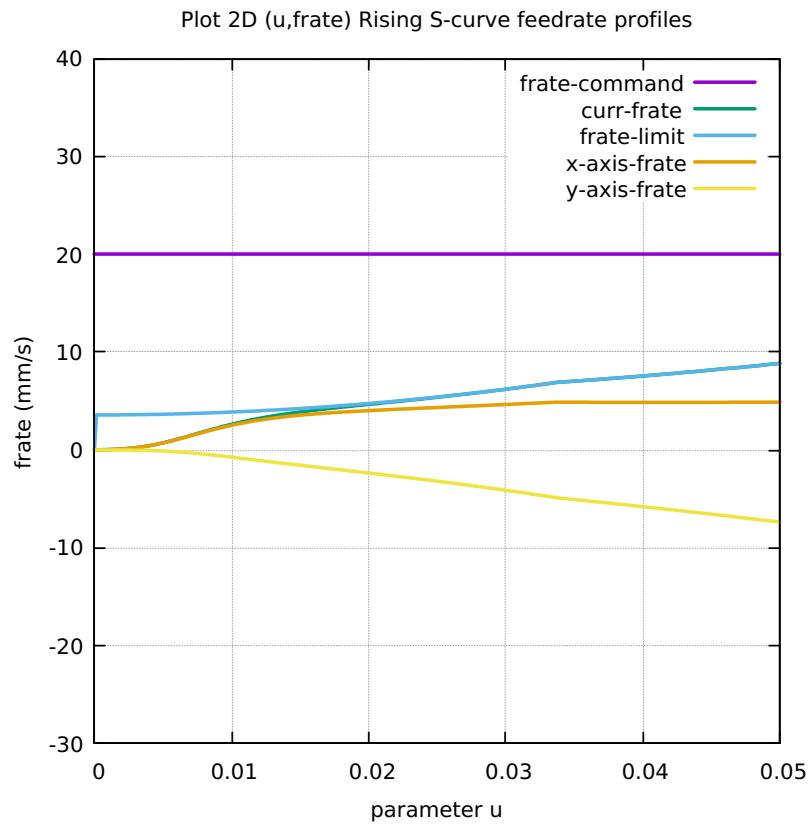


Figure 65: Ellipse FC20 Nominal Falling S-Curve Profile

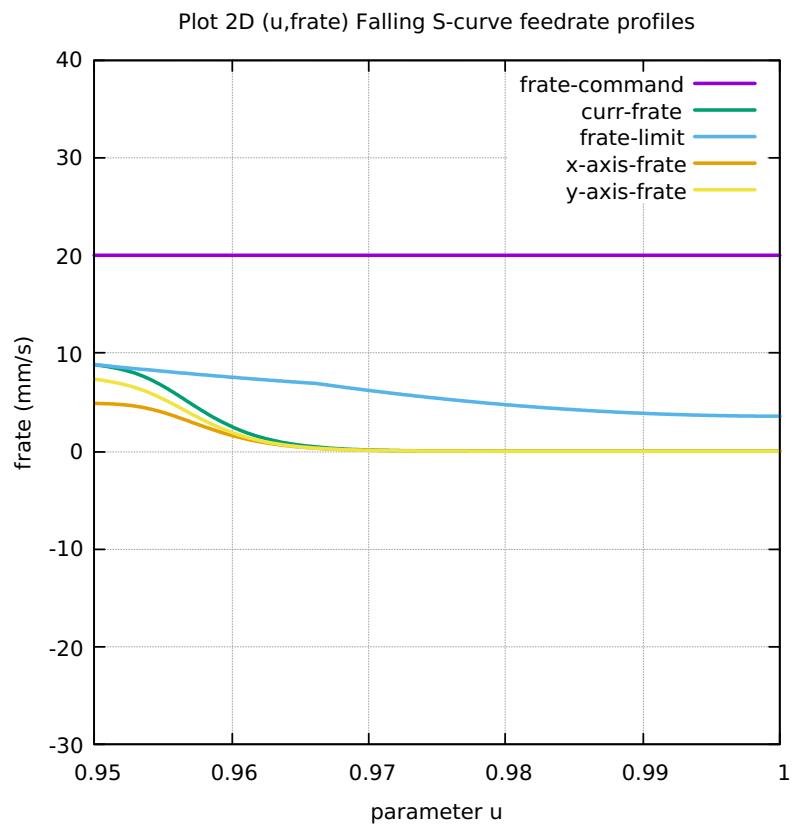


Figure 66: Ellipse FC10 Colored Feedrate Profile data ngcode

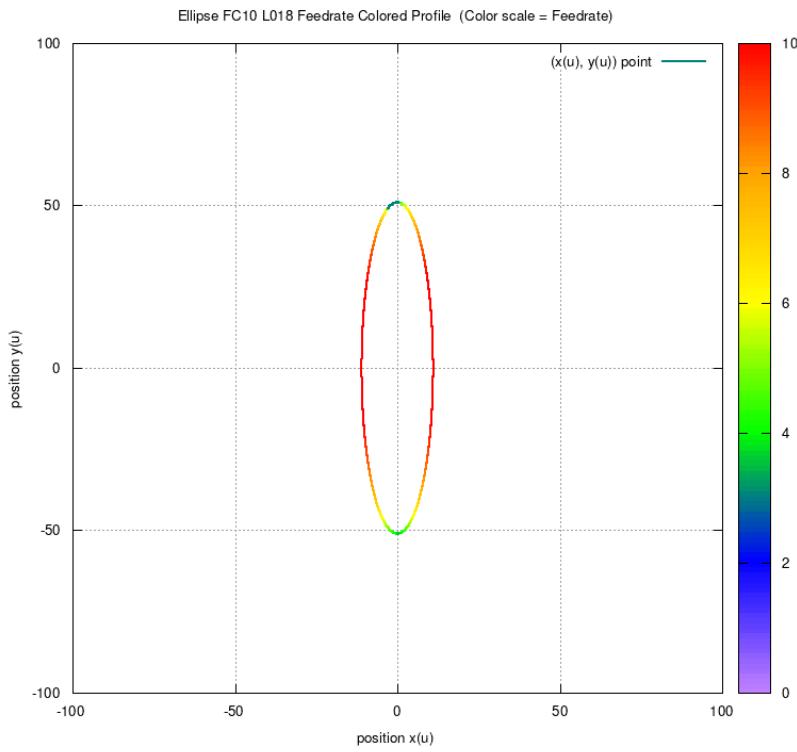


Figure 67: Ellipse FC20 Colored Feedrate Profile data ngcode

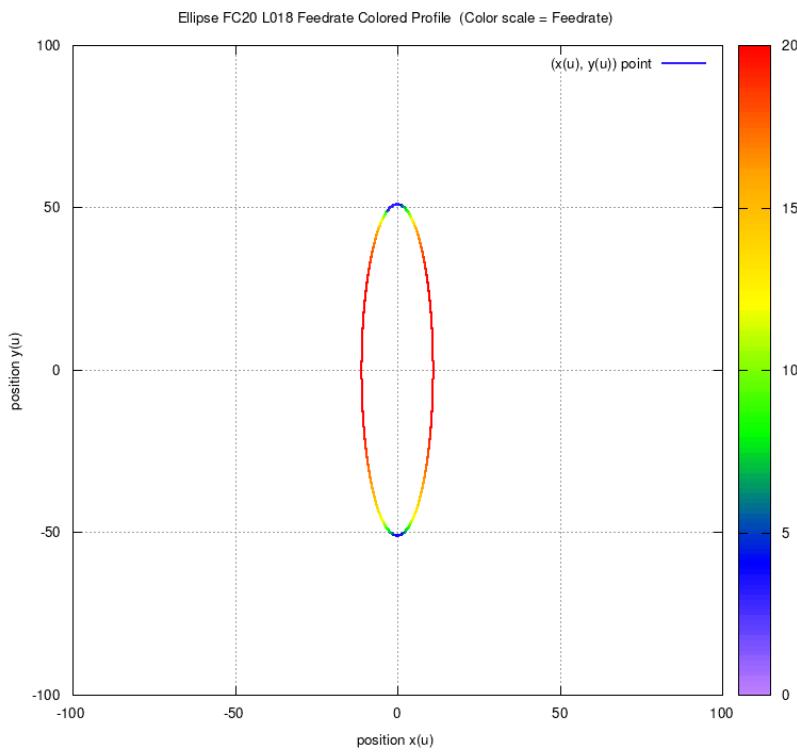


Figure 68: Ellipse FC30 Colored Feedrate Profile data ngcode

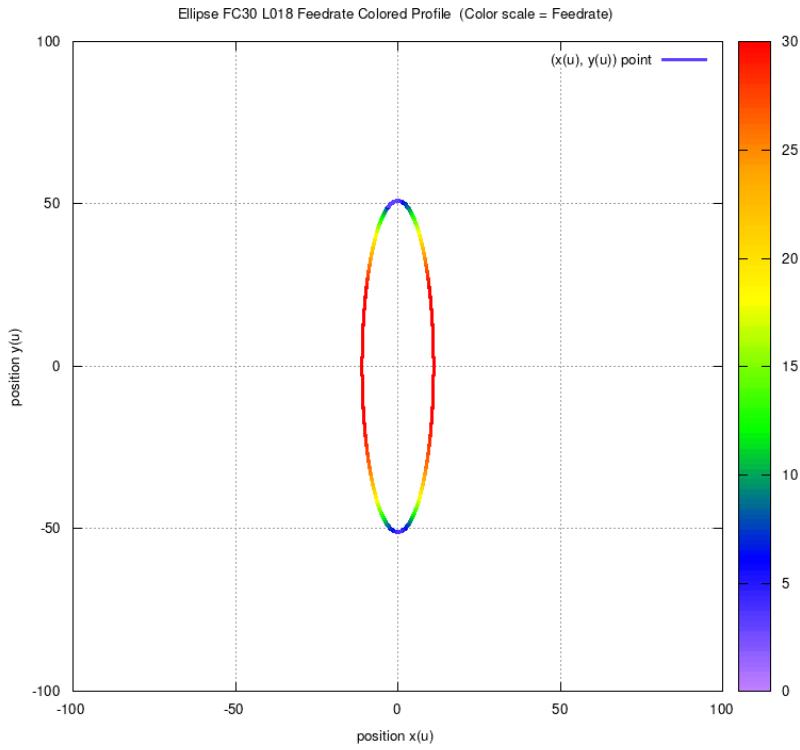


Figure 69: Ellipse FC40 Colored Feedrate Profile data ngcode

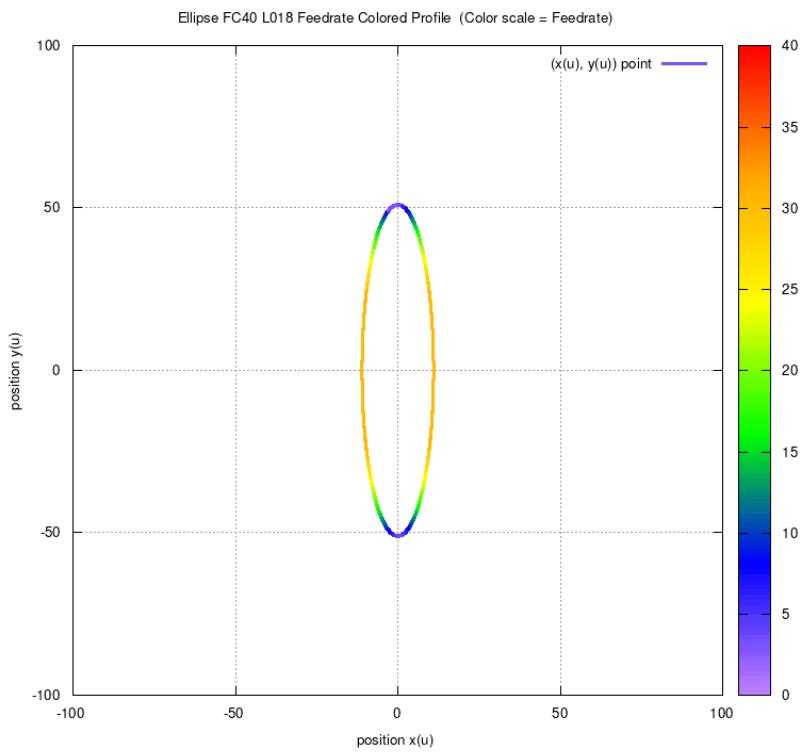


Figure 70: Ellipse FC10 Tangential Acceleration

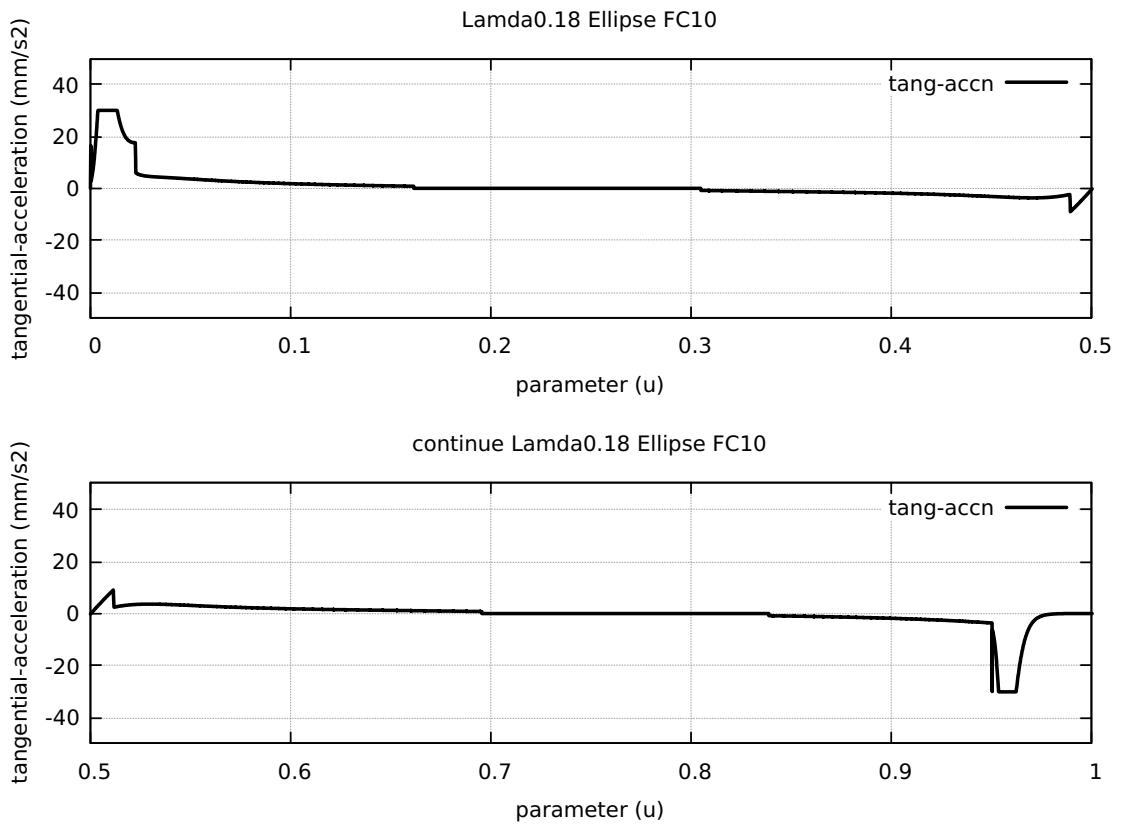


Figure 71: Ellipse FC20 Tangential Acceleration

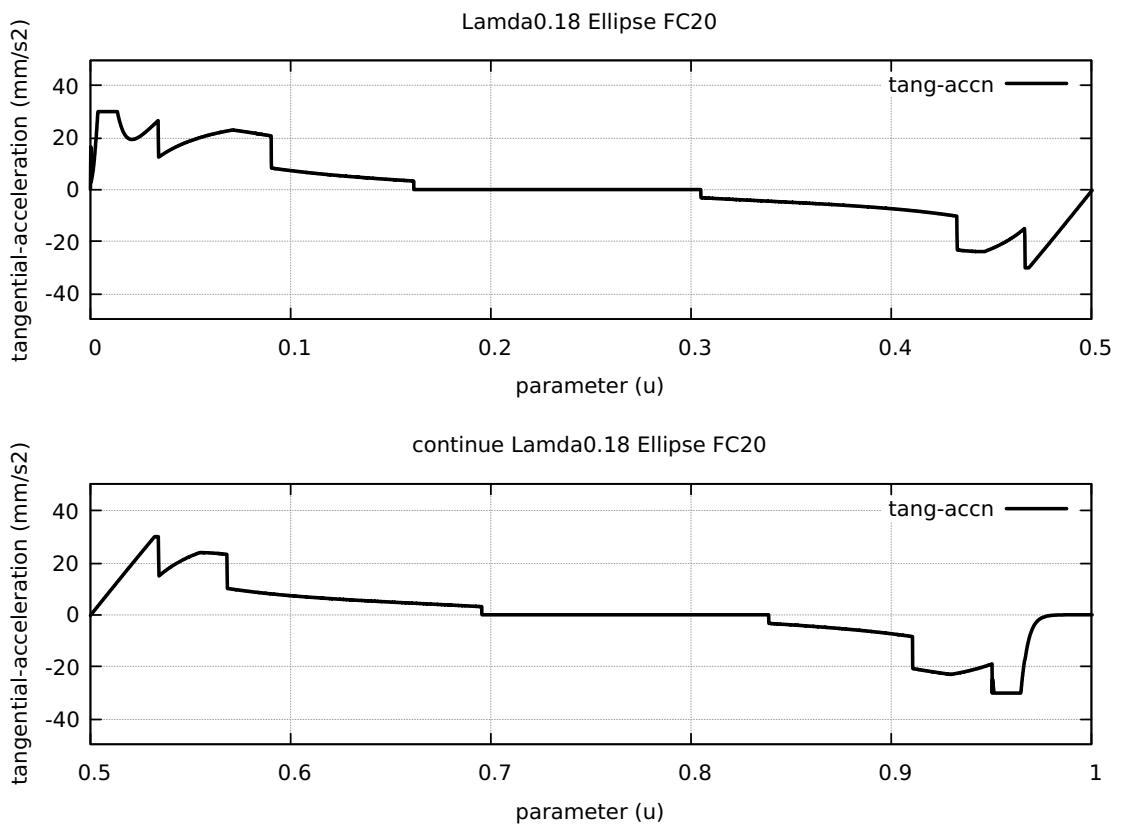


Figure 72: Ellipse FC30 Tangential Acceleration

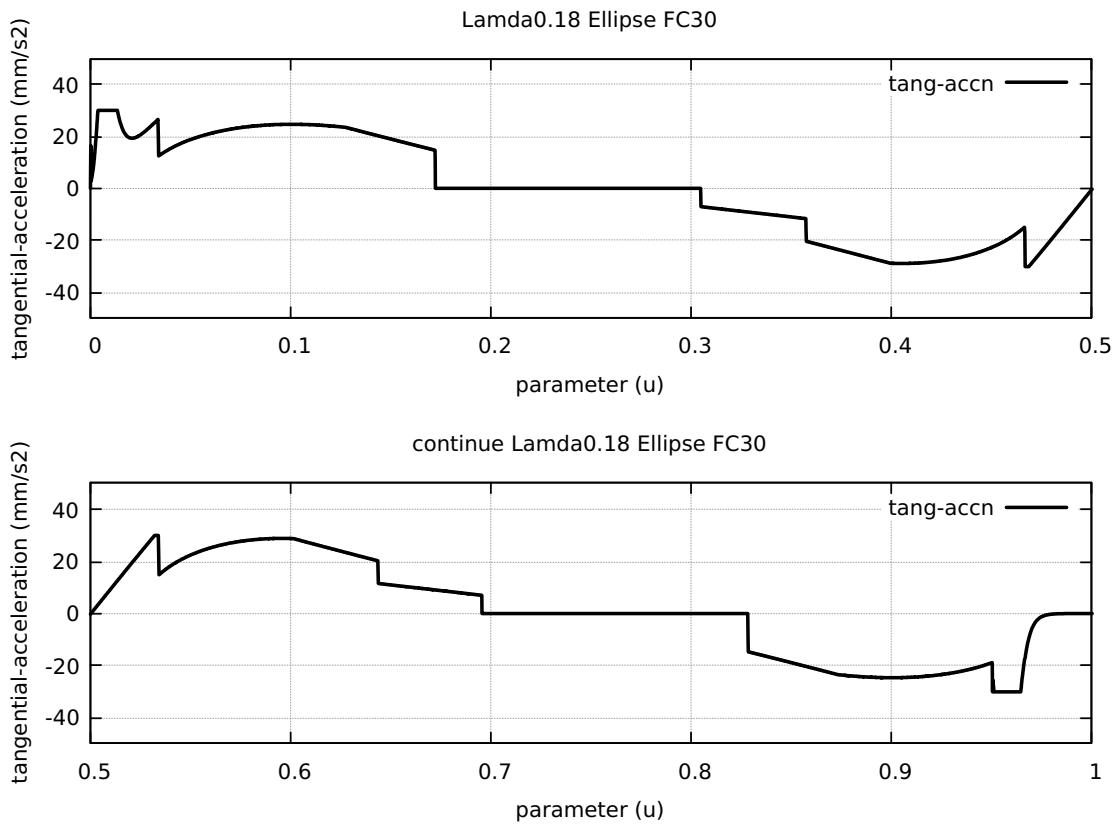


Figure 73: Ellipse FC40 Tangential Acceleration

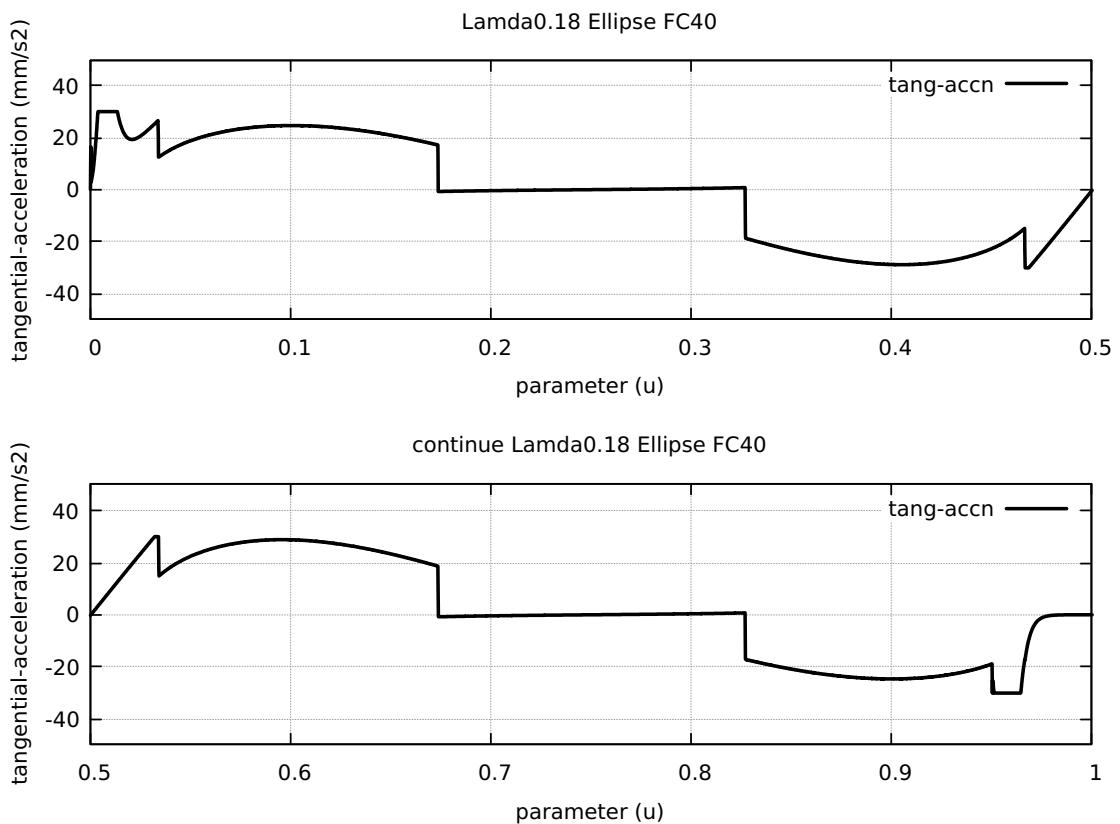


Figure 74: Ellipse FC20 Nominal Separation NAL and NCL

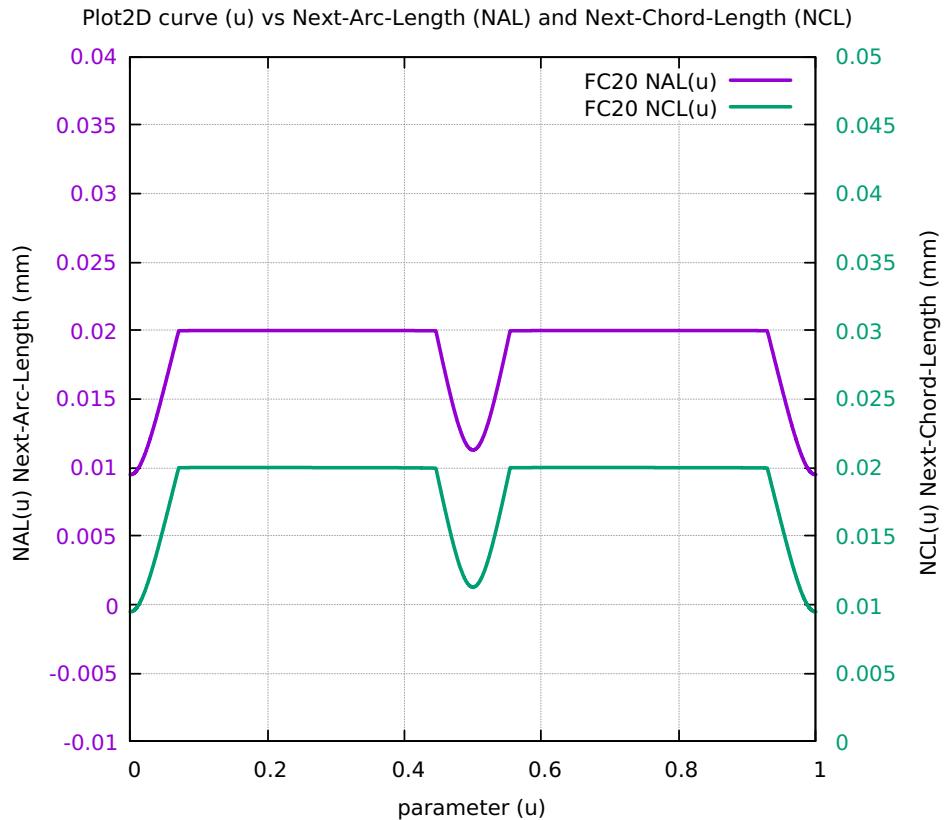


Figure 75: Ellipse Difference SAL minus SCL for FC10 FC20 FC30 FC40

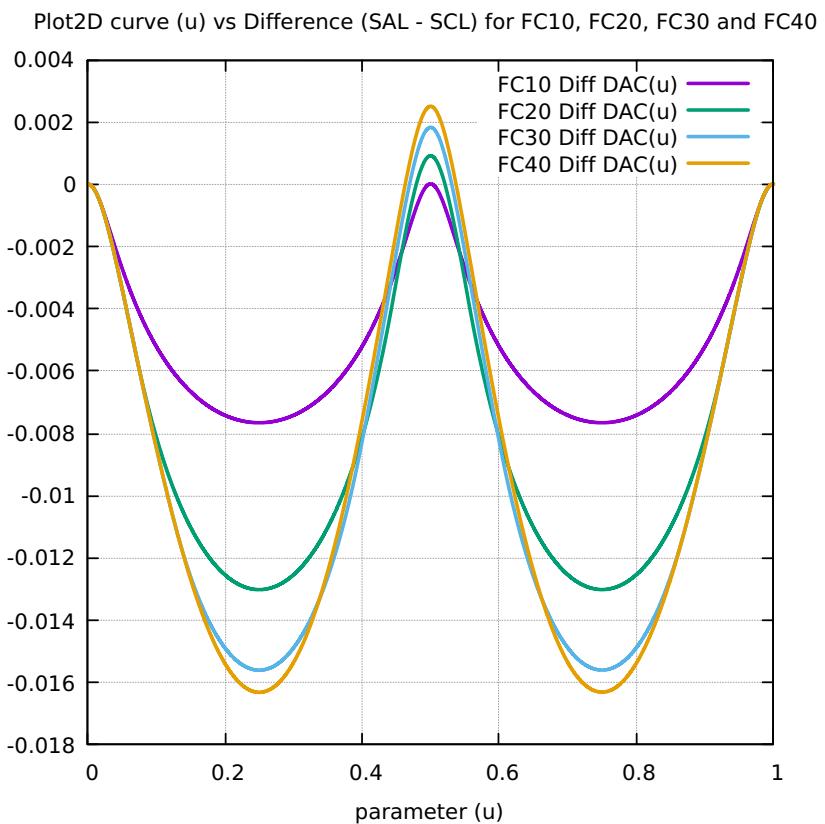


Figure 76: Ellipse FC10 FrateCmd CurrFrate X-Frate Y-Frate

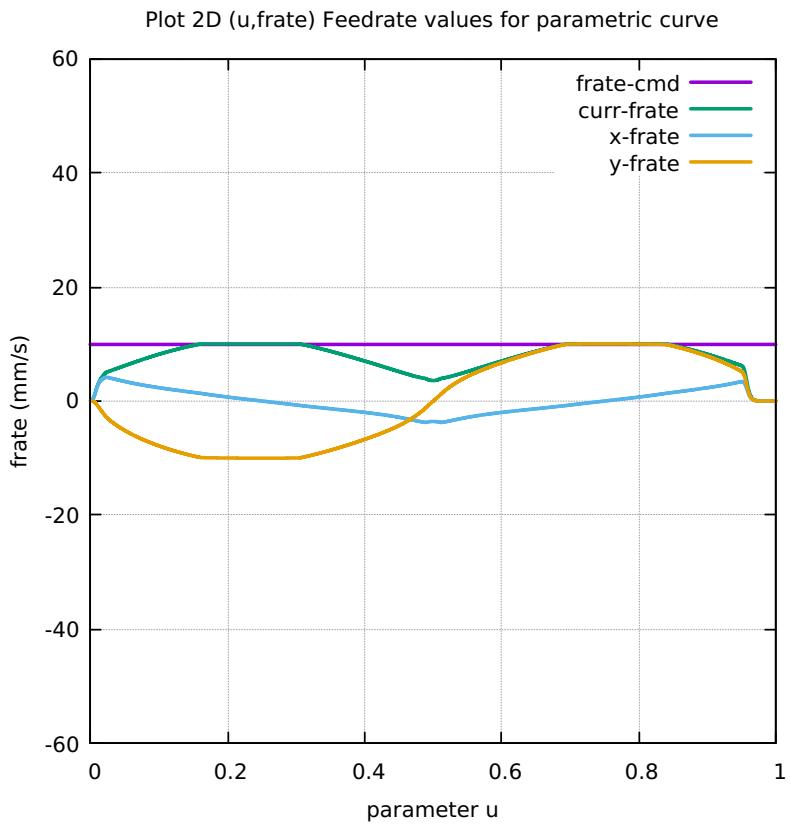


Figure 77: Ellipse FC20 FrateCmd CurrFrate X-Frate Y-Frate

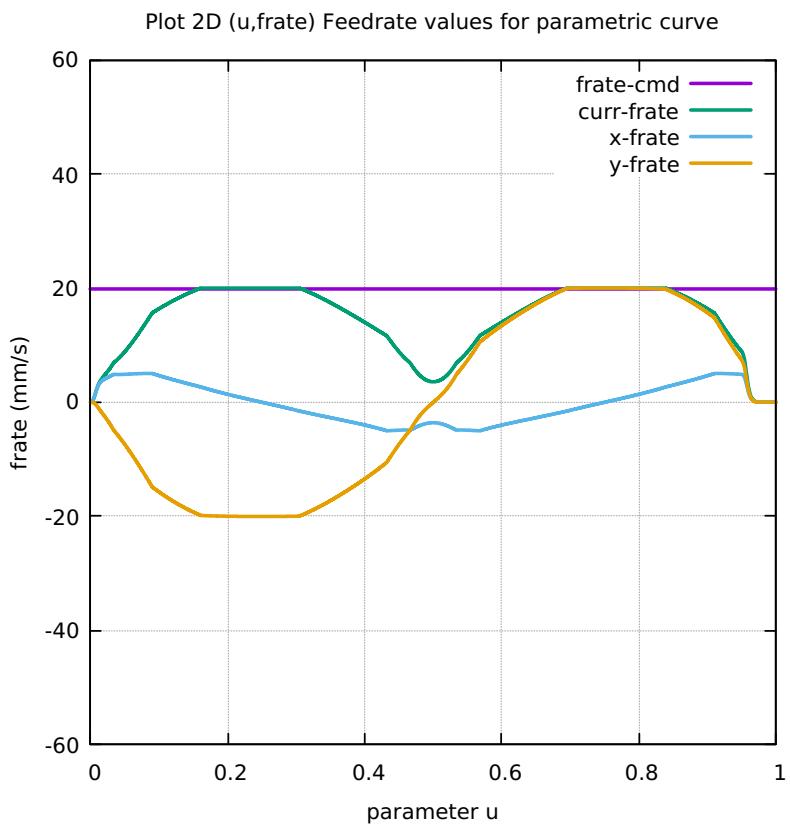


Figure 78: Ellipse FC30 FrateCmd CurrFrate X-Frate Y-Frate

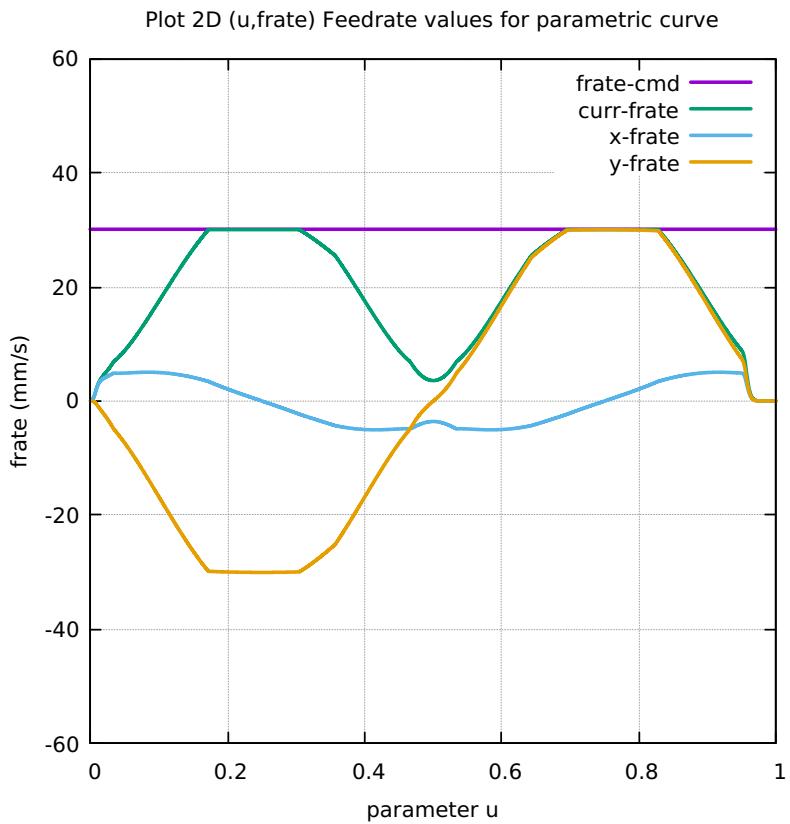


Figure 79: Ellipse FC40 FrateCmd CurrFrate X-Frate Y-Frate

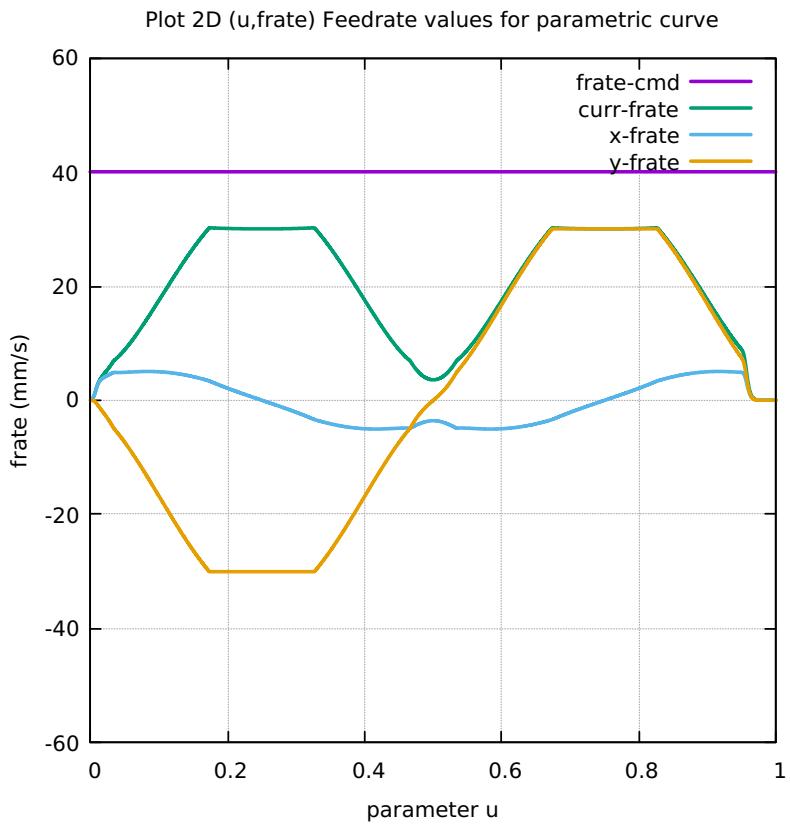


Figure 80: Ellipse FC10 Four Components FeedrateLimit

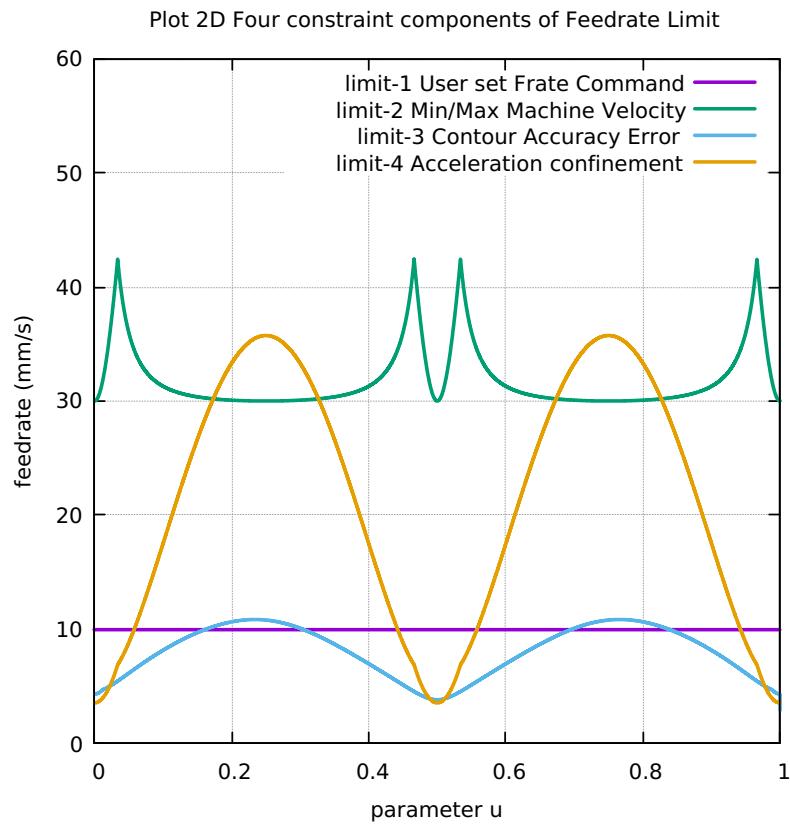


Figure 81: Ellipse FC20 Four Components FeedrateLimit

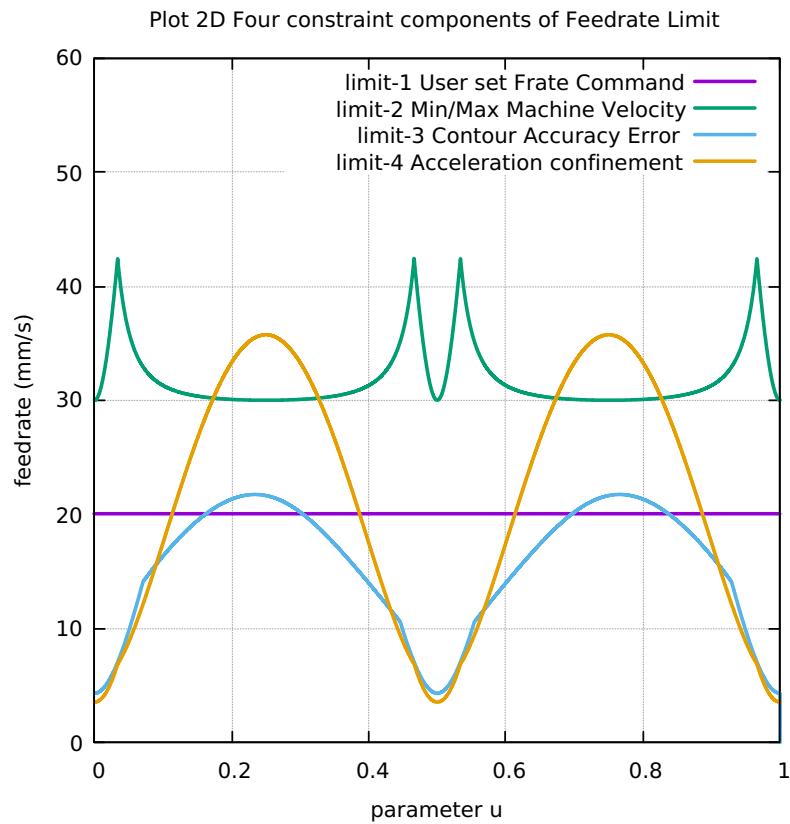


Figure 82: Ellipse FC30 Four Components FeedrateLimit

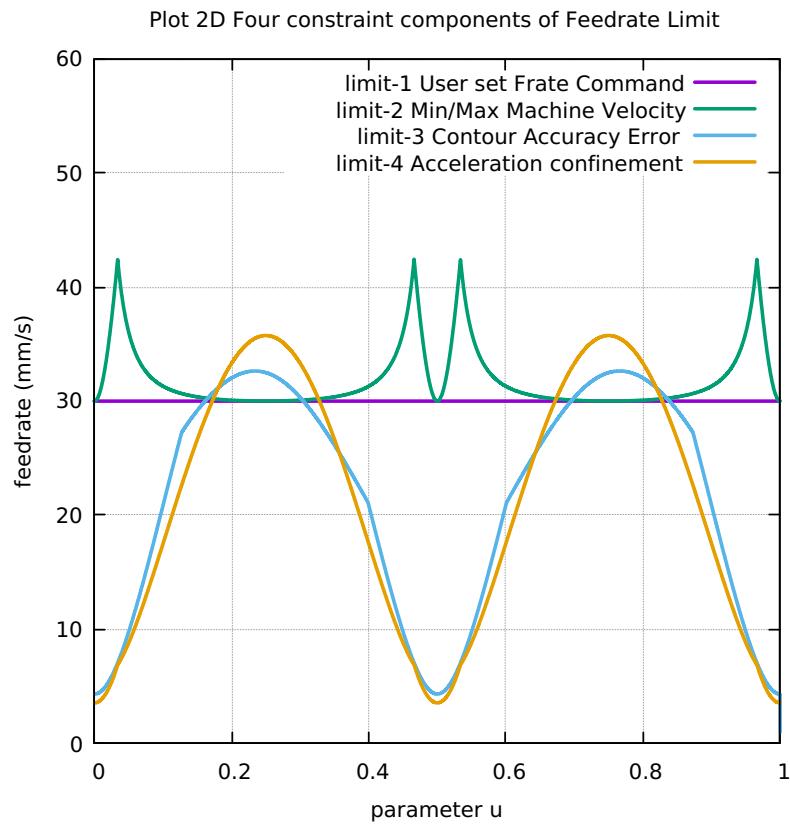


Figure 83: Ellipse FC40 Four Components FeedrateLimit

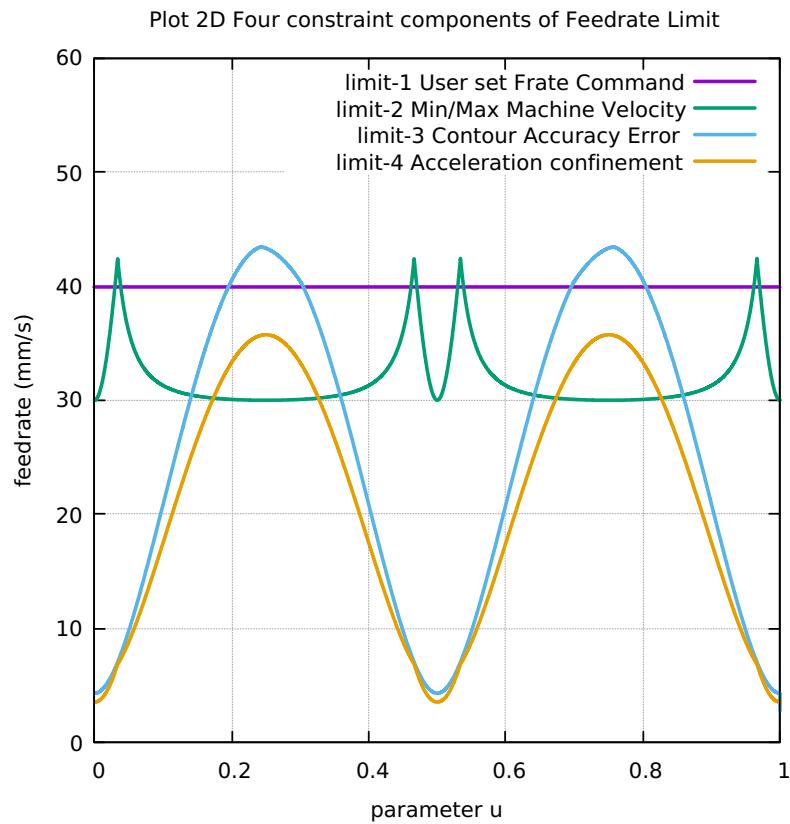


Figure 84: Ellipse Histogram Points FC10 FC20 FC30 FC40

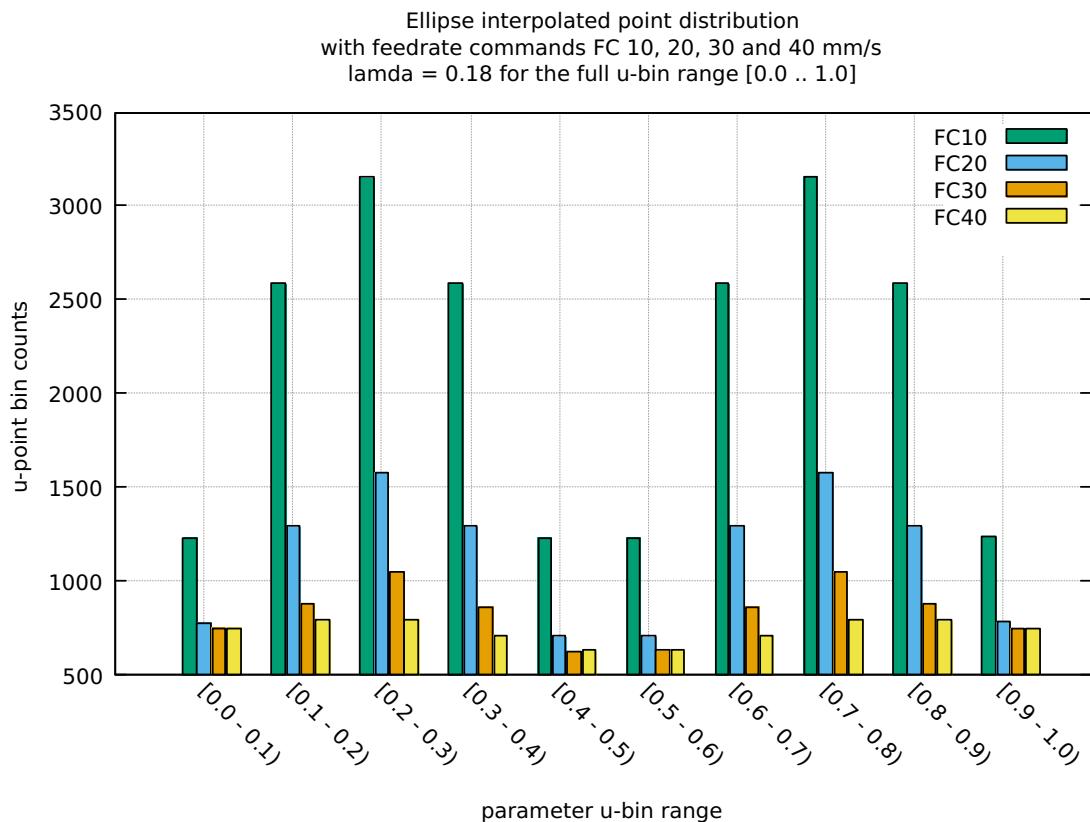


Table 4: Ellipse Table distribution of interpolated points

BINS	FC10	FC20	FC30	FC40
0.0 - 0.1	4964	2482	1654	1241
0.1 - 0.2	4964	2482	1655	1241
0.2 - 0.3	4964	2482	1655	1241
0.3 - 0.4	4964	2482	1655	1241
0.4 - 0.5	4964	2482	1655	1242
0.5 - 0.6	4964	2483	1655	1241
0.6 - 0.7	4964	2482	1655	1241
0.7 - 0.8	4964	2482	1655	1241
0.8 - 0.9	4964	2482	1654	1242
0.9 - 1.0	4965	2483	1656	1242
Tot Counts	49641	24822	16549	12413

Table 5: Ellipse Table FC10-20-30-40 Run Performance data

1	Curve Type	ELLIPSE	ELLIPSE	ELLIPSE
2	User Feedrate Command FC(mm/s)	FC10	FC20	FC40
3	User Lamda Acceleration Safety Factor	0.18	0.18	0.18
4	Total Interpolated Points (TIP)	21575	11296	8338
5	Total Sum-Chord-Error (SCE) (mm)	2.990951697698E-03	5.148661124856E-03	6.561982225601E-03
6	Ratio 1 = (SCE/TIP) = Chord-Error/Point	1.386368637108E-07	4.558354249540E-07	7.870915467915E-07
7	Total Sum-Arc-Length (SAL) (mm)	2.156436635306E+02	2.156499394203E+02	2.156478495089E+02
8	Total Sum-Chord-Length (SCL) (mm)	2.156436625529E+02	2.156499358521E+02	2.156478426167E+02
9	Difference = (SAL - SCL) (mm)	9.777115224097E-07	3.568242959773E-06	6.892209597709E-06
10	Percentage Difference = (SAL - SCL)/SAL	4.533921870933E-07	1.654645936542E-06	3.196048378596E-06
11	Ratio 2 = (SCE/SCL) = Chord Error/Chord-Length	1.386987988559E-05	2.387508767166E-05	3.042915776933E-05
12	Total Sum-Arc-Theta (SAT) (rad)	6.280515723261E+00	6.283169115421E+00	6.282291919271E+00
13	Total Sum-Arc-Area (SAA) (mm2)	5.185017589366E-05	1.275076043292E-04	1.849039783361E-04
14	Ratio 3 = (SAA/SCL) = Arc-Area/Chord-Length	1.386987988559E-05	2.387508767166E-05	3.042915776933E-05
15	Average-Chord-Error (ACE) (mm)	1.386368637108E-07	4.558354249540E-07	7.870915467915E-07
16	Average-Arc-Length (AAL) (mm)	9.995534603255E-03	1.909251345023E-02	2.586636074235E-02
17	Average-Chord-Length (ACL) (mm)	9.995534557936E-03	1.909251313431E-02	2.586635991565E-02
18	Average-Arc-Theta (AAT) (rad)	2.911150330611E-04	5.562788061462E-04	7.535434711852E-04
19	Average-Arc-Area (AAA) (mm2)	2.403364044390E-09	1.128885385827E-08	2.217871876407E-08
20	Algorithm actual runtime on computer (ART) (s)	7.13439876	7.633051722	11.041426063
				16.776571251

.5 APPENDIX TEARDROP CURVE

- .5.1 Plot of Teardrop curve [85]**
- .5.2 Teardrop Radius of Curvature [86]**
- .5.3 Teardrop Validation in LinuxCNC [87]**
- .5.4 Teardrop Direction of Travel 3D [88]**
- .5.5 Teardrop First and Second Order Taylor's Approx [89]**
- .5.6 Teardrop First minus Second Order Taylor's Approx [90]**
- .5.7 Teardrop Separate First Second Order Taylor's Approx [91]**
- .5.8 Teardrop Separation SAL and SCL [92]**
- .5.9 Teardrop Chord-error in close view 2 scales [93]**
- .5.10 Teardrop Four Components Feedrate Limit [94]**
- .5.11 Teardrop FrateCommand FrateLimit and Curr-Frate [95]**
- .5.12 Teardrop FeedRateLimit minus CurrFeedRate [96]**
- .5.13 Teardrop FC20-Nominal X and Y Feedrate Profiles [97]**
- .5.14 Teardrop FC20 Nominal Tangential Acceleration [98]**
- .5.15 Teardrop FC20 Nominal Rising S-Curve Profile [99]**
- .5.16 Teardrop FC20 Nominal Falling S-Curve Profile [100]**
- .5.17 Teardrop FC10 Colored Feedrate Profile data ngcode [101]**
- .5.18 Teardrop FC20 Colored Feedrate Profile data ngcode [102]**

- .5.19 Teardrop FC30 Colored Feedrate Profile data ngcode [103]
- .5.20 Teardrop FC40 Colored Feedrate Profile data ngcode [104]
- .5.21 Teardrop FC10 Tangential Acceleration [105]
- .5.22 Teardrop FC20 Tangential Acceleration [106]
- .5.23 Teardrop FC30 Tangential Acceleration [107]
- .5.24 Teardrop FC40 Tangential Acceleration [108]
- .5.25 Teardrop FC20 Nominal Separation NAL and NCL [109]
- .5.26 Teardrop SAL minus SCL for FC10 FC20 FC30 FC40 [110]
- .5.27 Teardrop FC10 FrateCmd CurrFrate X-Frate Y-Frate [111]
- .5.28 Teardrop FC20 FrateCmd CurrFrate X-Frate Y-Frate [112]
- .5.29 Teardrop FC30 FrateCmd CurrFrate X-Frate Y-Frate [113]
- .5.30 Teardrop FC40 FrateCmd CurrFrate X-Frate Y-Frate [114]
- .5.31 Teardrop FC10 Four Components FeedrateLimit [115]
- .5.32 Teardrop FC20 Four Components FeedrateLimit [116]
- .5.33 Teardrop FC30 Four Components FeedrateLimit [117]
- .5.34 Teardrop FC40 Four Components FeedrateLimit [118]
- .5.35 Teardrop Histogram Points FC10 FC20 FC30 FC40 [119]
- .5.36 Teardrop Table distribution of interpolated points [6]
- .5.37 Teardrop Table FC10-20-30-40 Run Performance data [7]

Figure 85: Plot of Teardrop curve

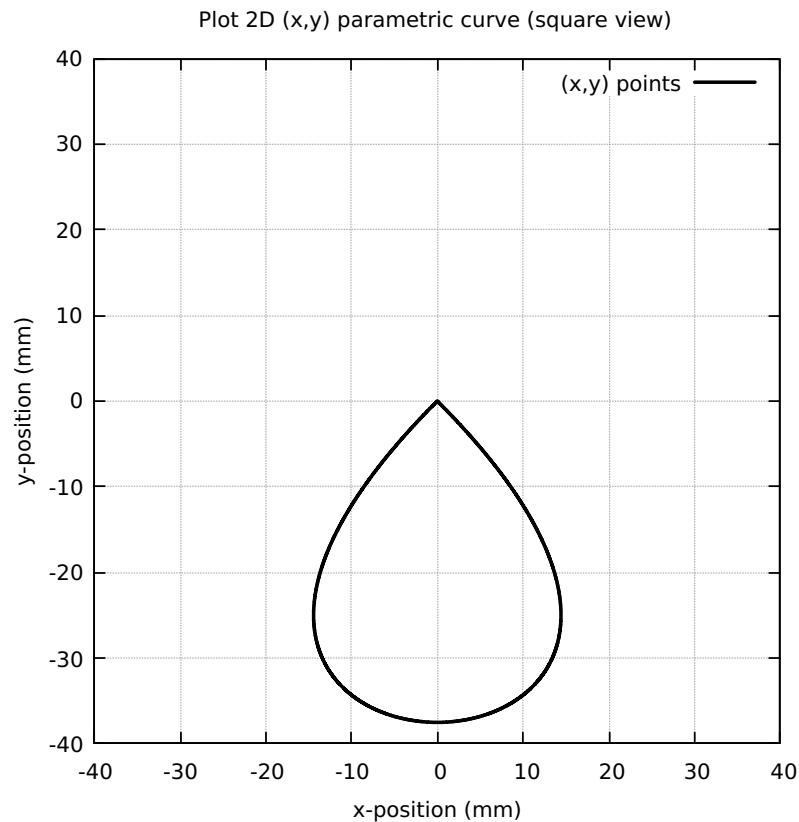


Figure 86: Teardrop Radius of Curvature

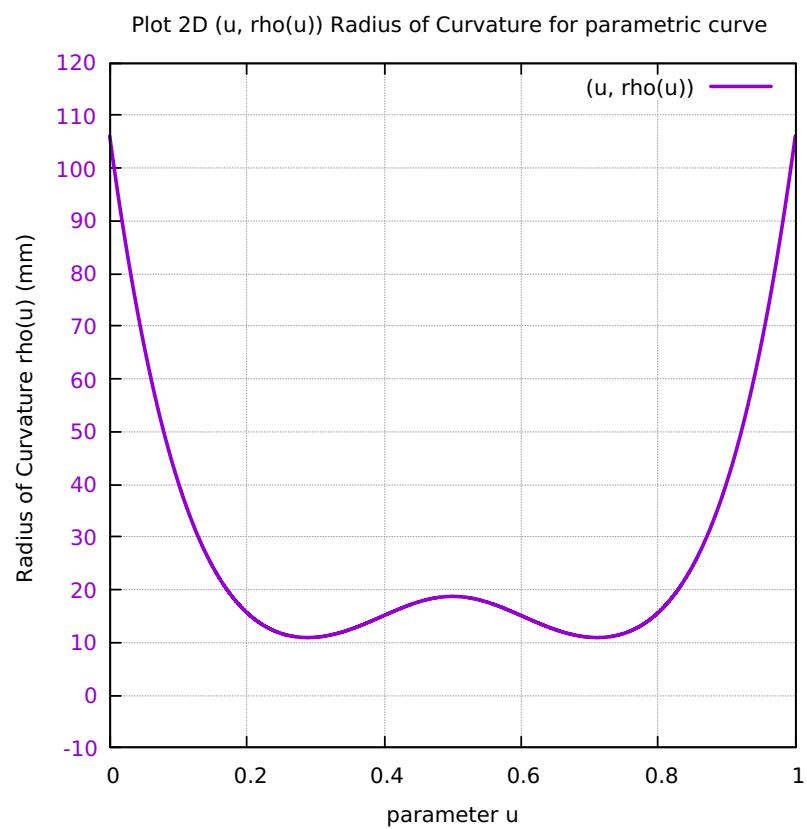


Figure 87: Teardrop Validation in LinuxCNC

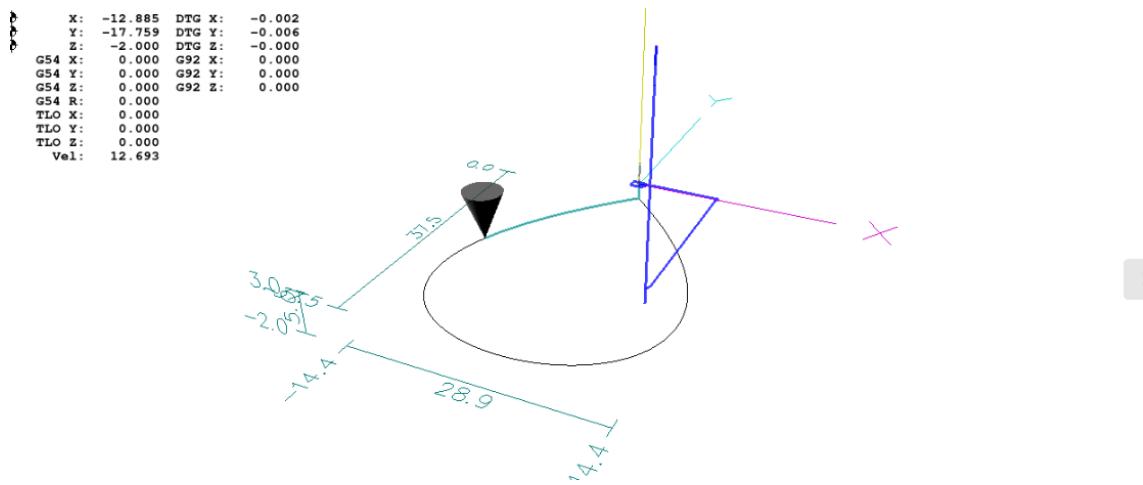


Figure 88: Teardrop Direction of Travel 3D

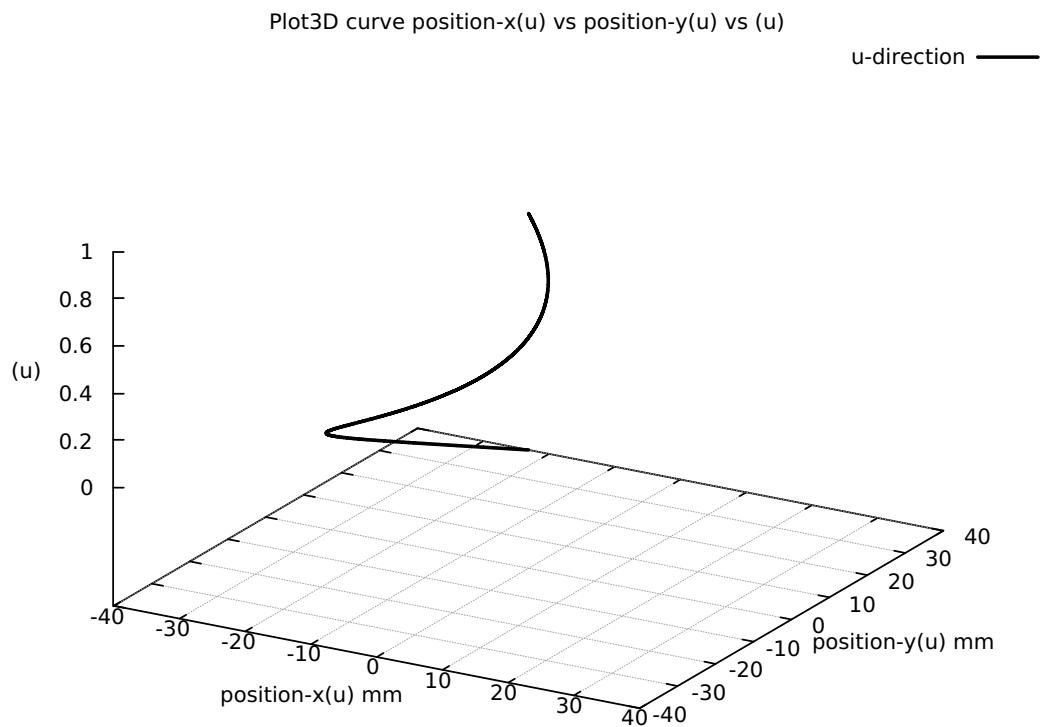


Figure 89: Teardrop First and Second Order Taylor's Approximation

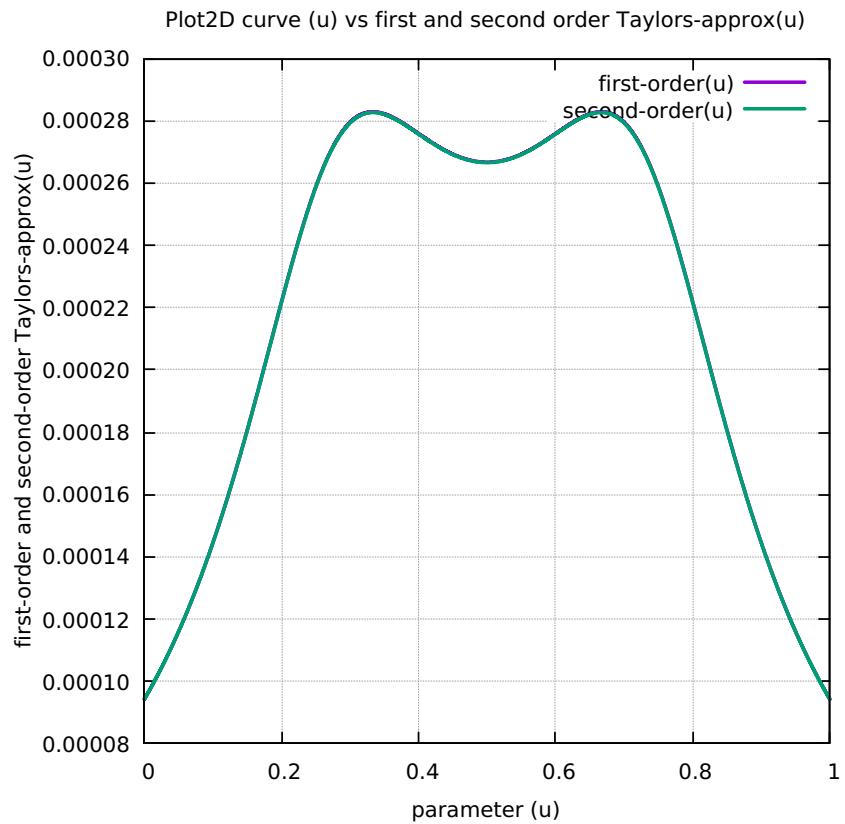


Figure 90: Teardrop First minus Second Order Taylor's Approximation

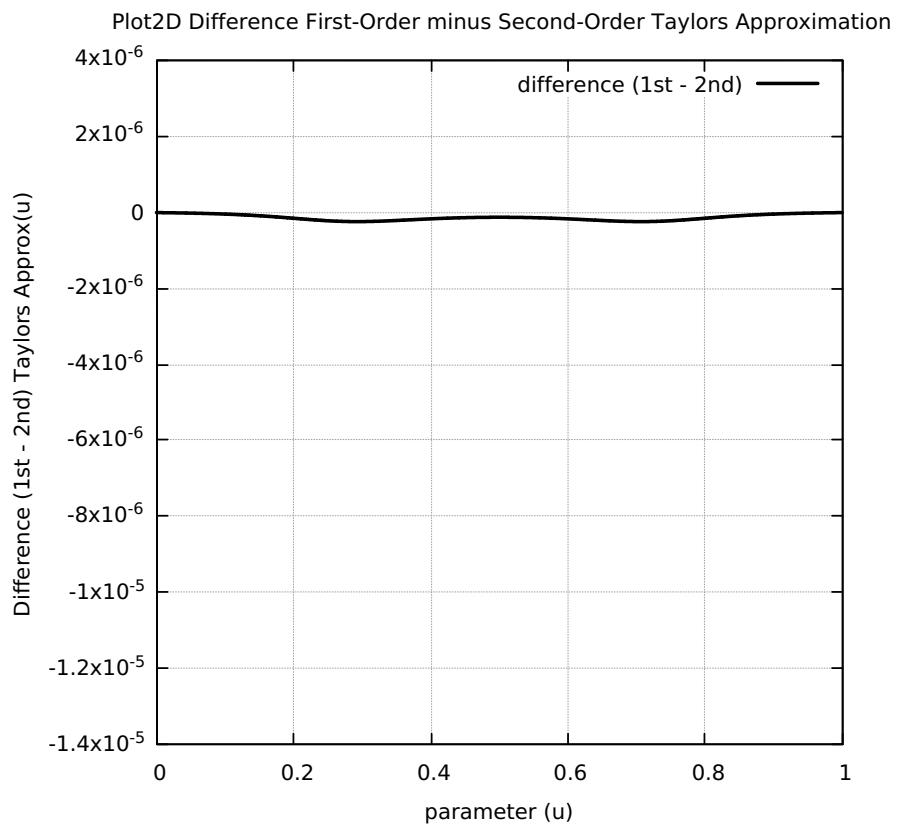


Figure 91: Teardrop Separation First and Second Order Taylor's Approximation

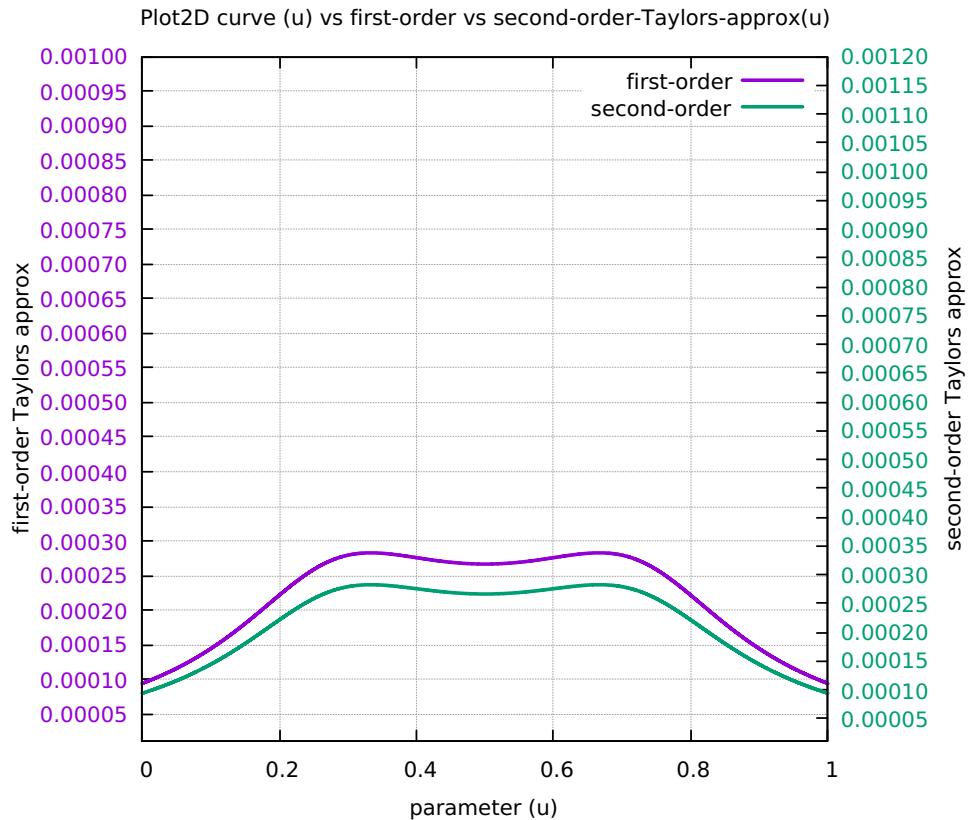


Figure 92: Teardrop Separation SAL and SCL

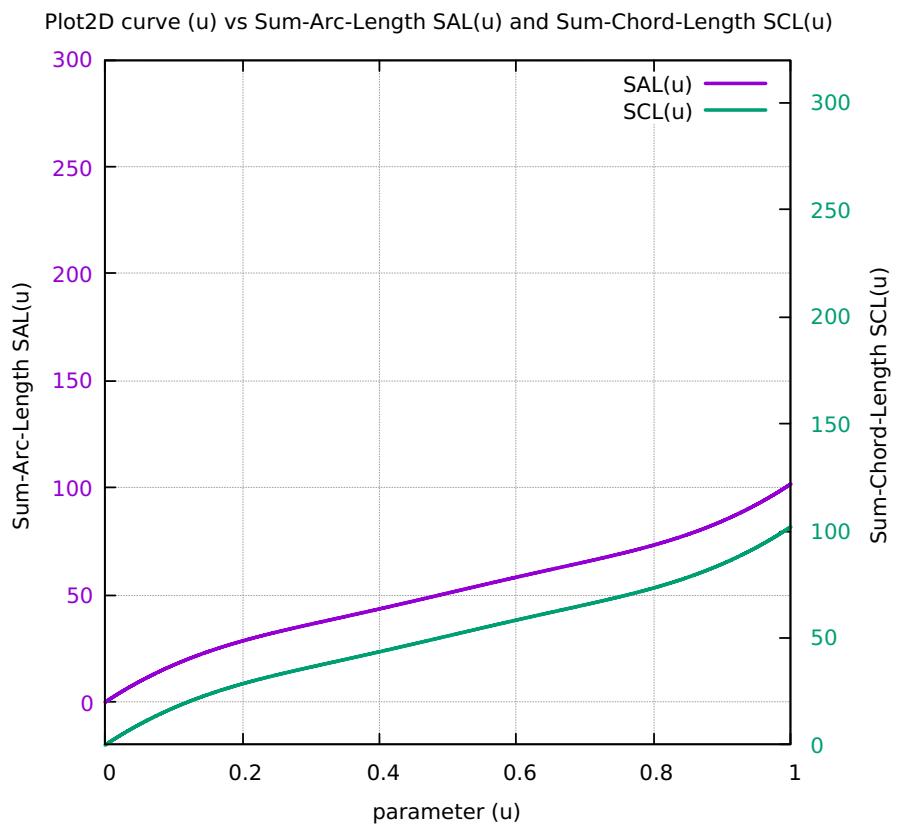


Figure 93: Teardrop Chord-error in close view 2 scales

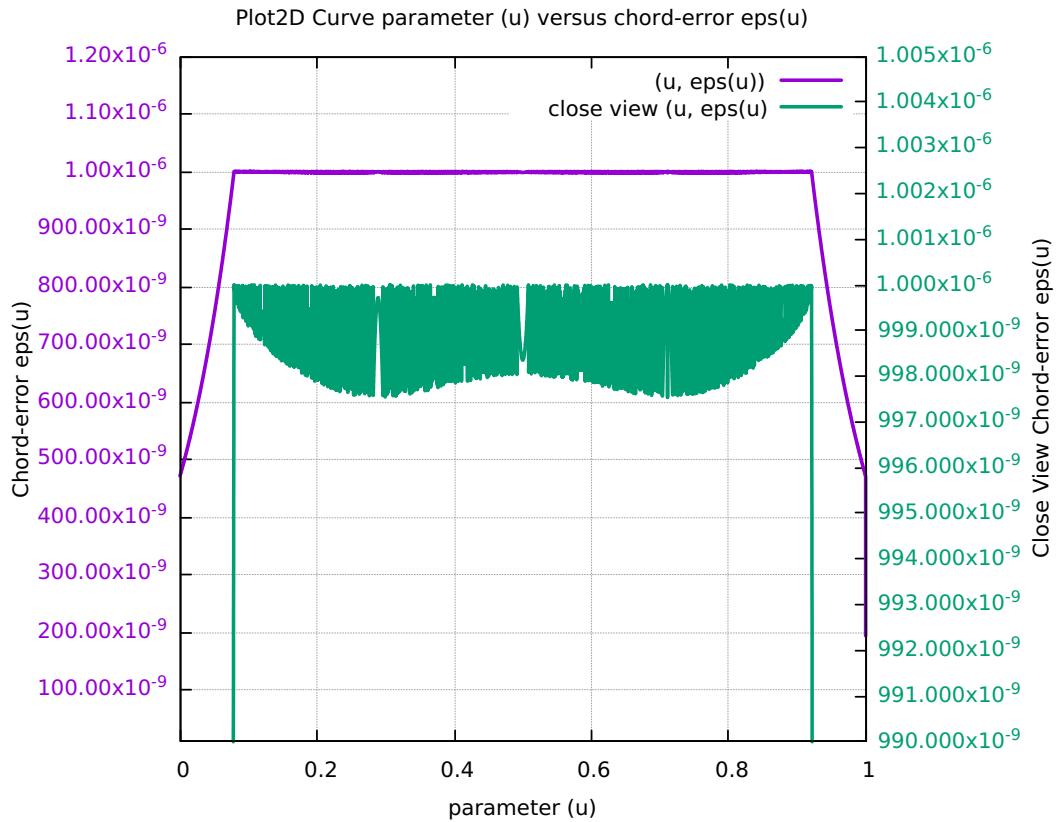


Figure 94: Teardrop Four Components Feedrate Limit

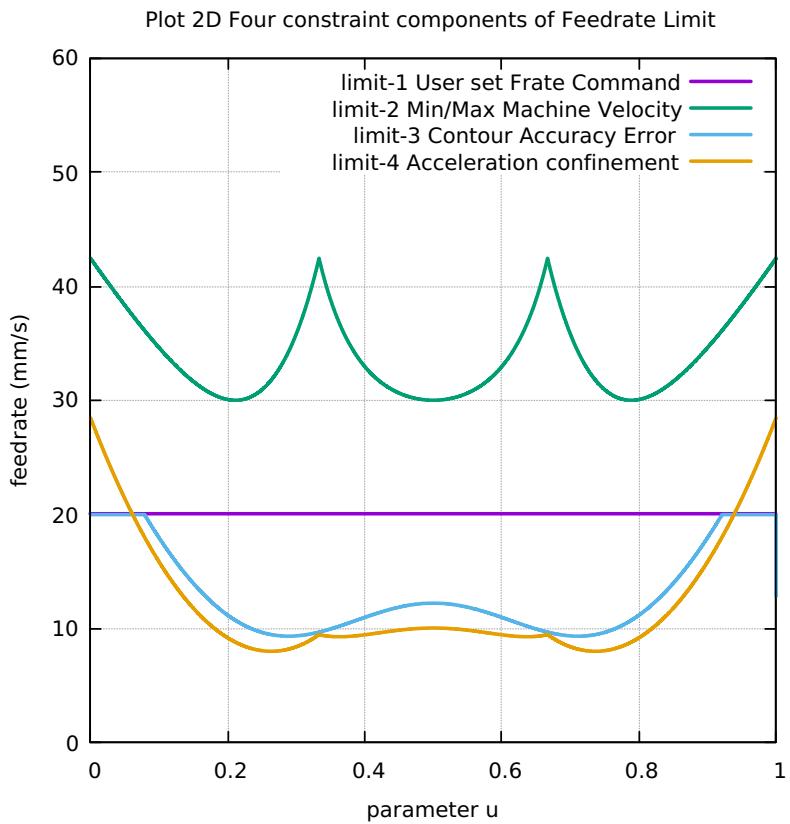


Figure 95: Teardrop FrateCommand FrateLimit and Curr-Frate

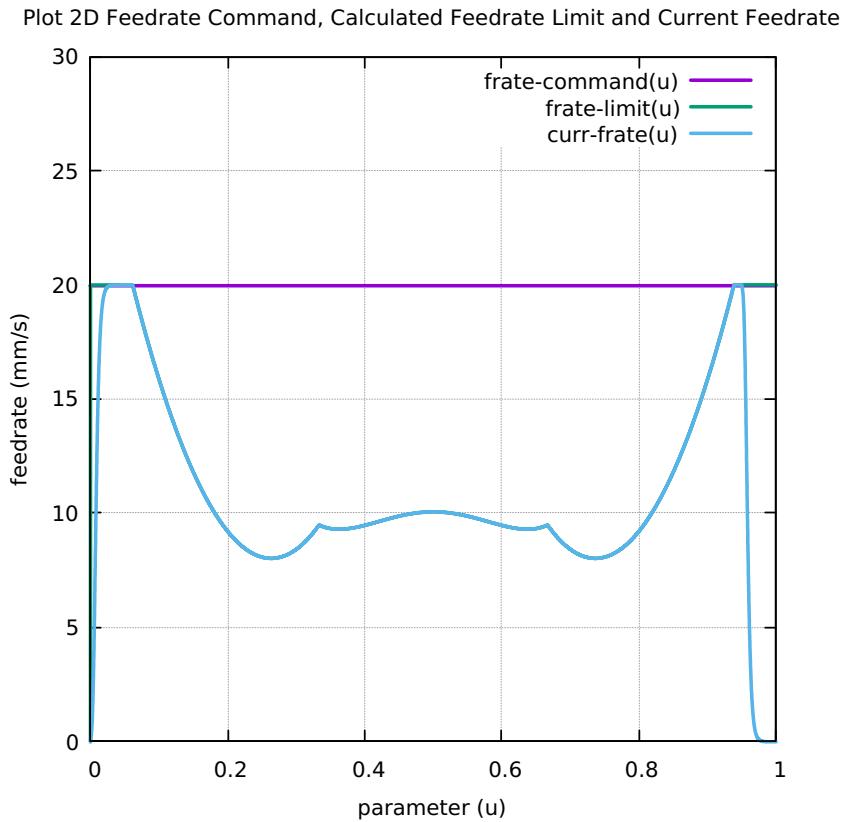


Figure 96: Teardrop FeedRateLimit minus CurrFeedRate

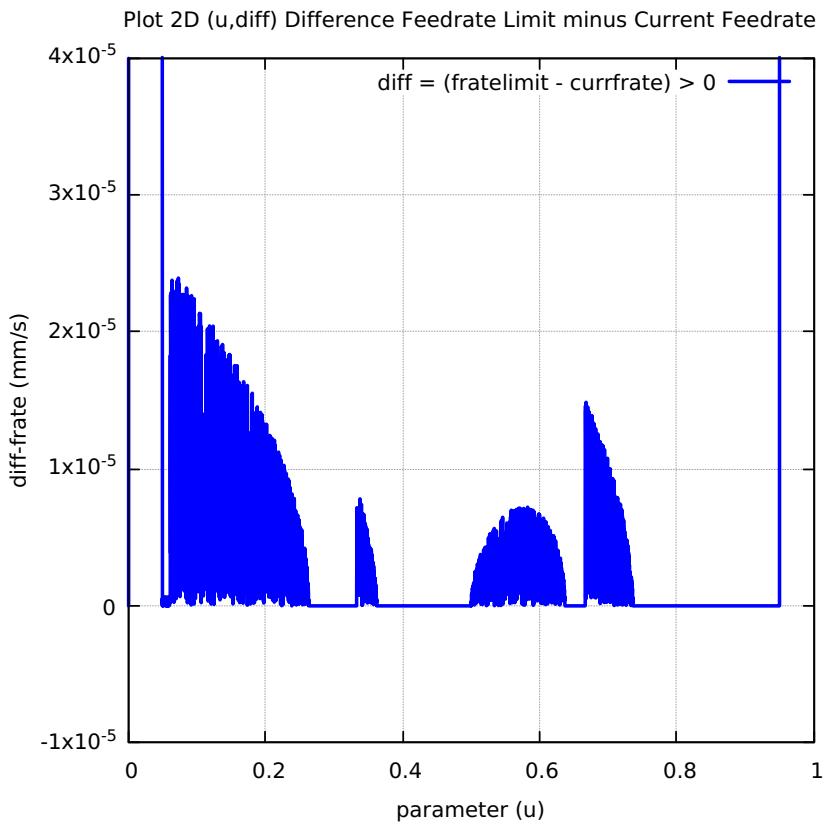


Figure 97: Teardrop FC20-Nominal X and Y Feedrate Profiles

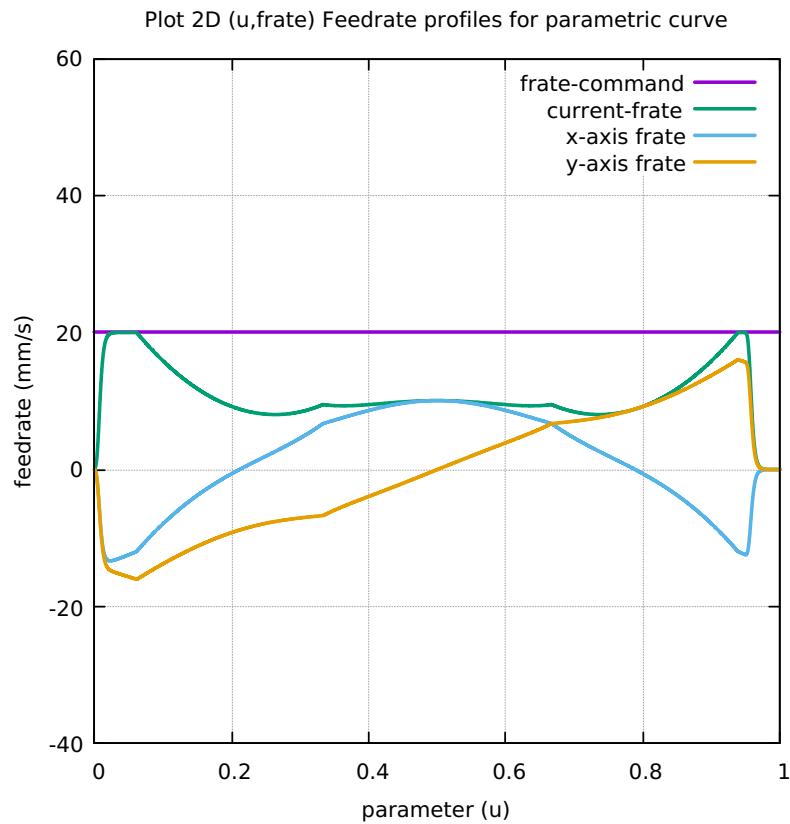


Figure 98: Teardrop FC20 Nominal Tangential Acceleration

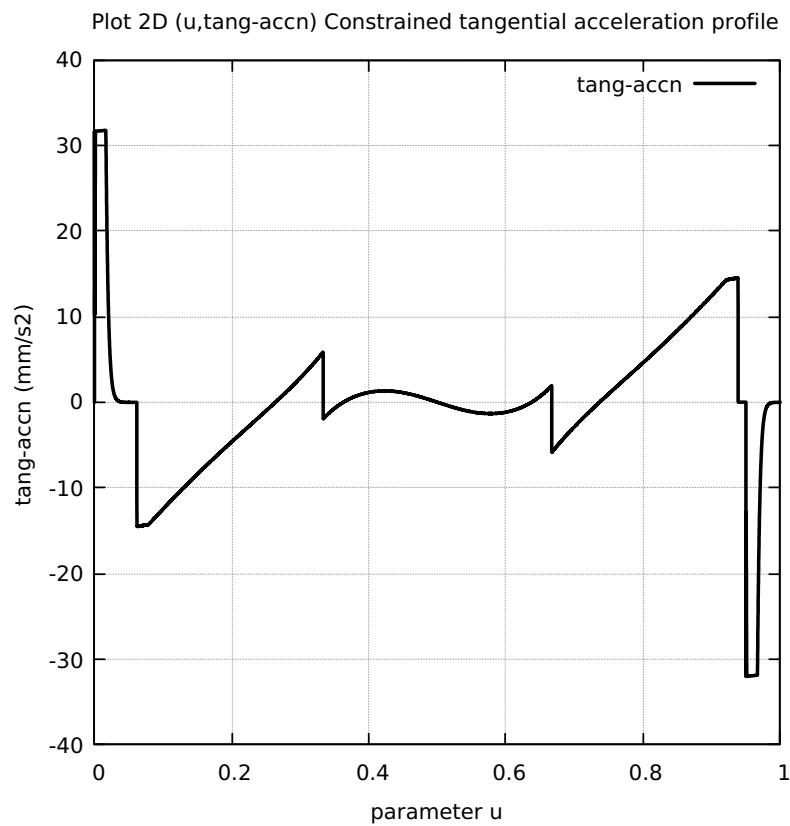


Figure 99: Teardrop FC20 Nominal Rising S-Curve Profile

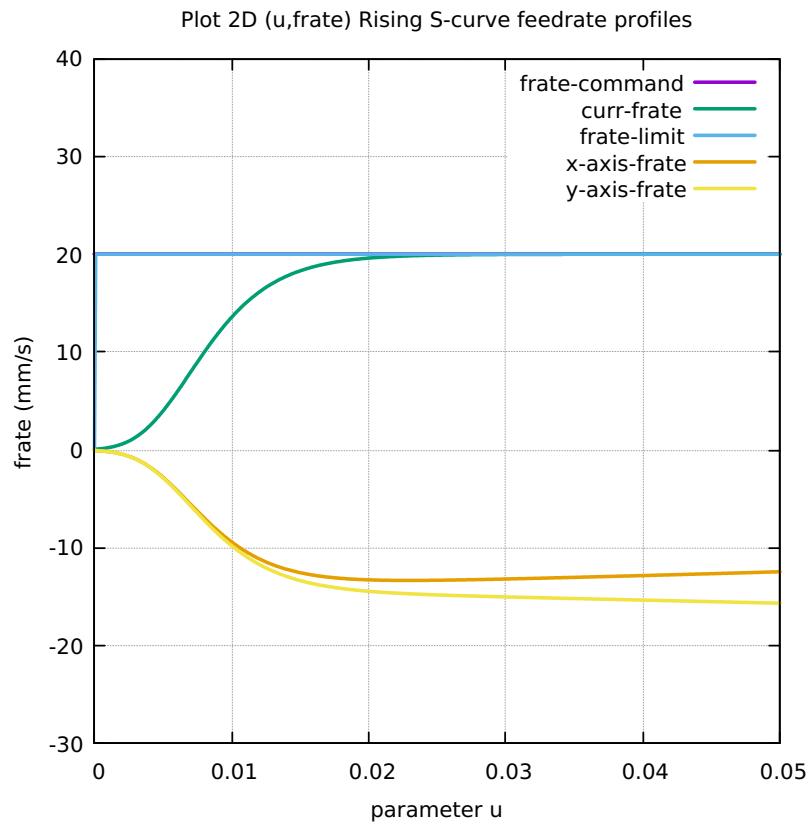


Figure 100: Teardrop FC20 Nominal Falling S-Curve Profile

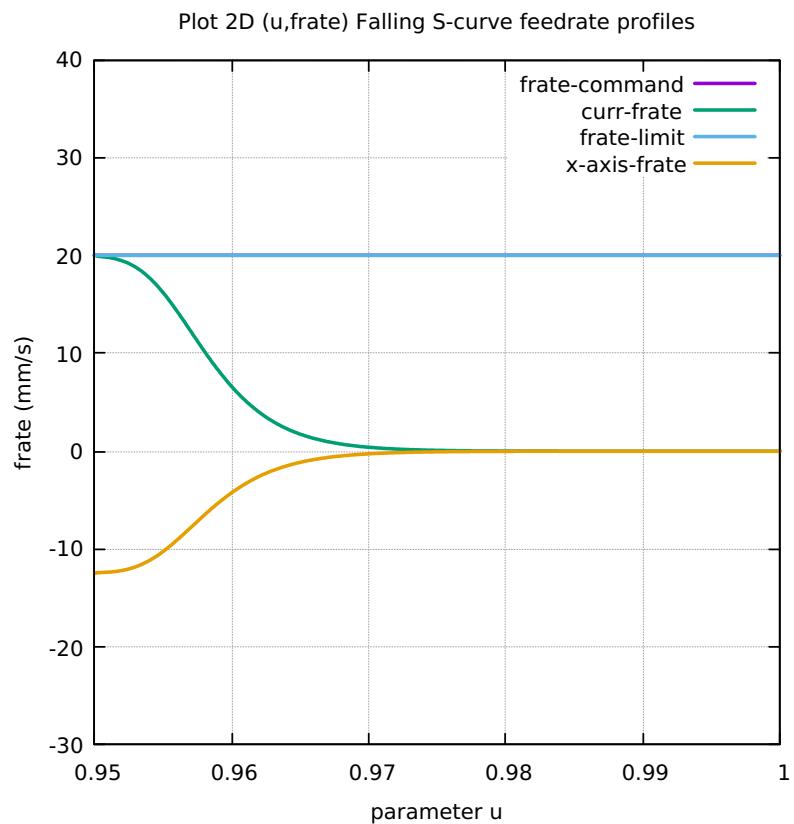


Figure 101: Teardrop FC10 Colored Feedrate Profile data ngcode

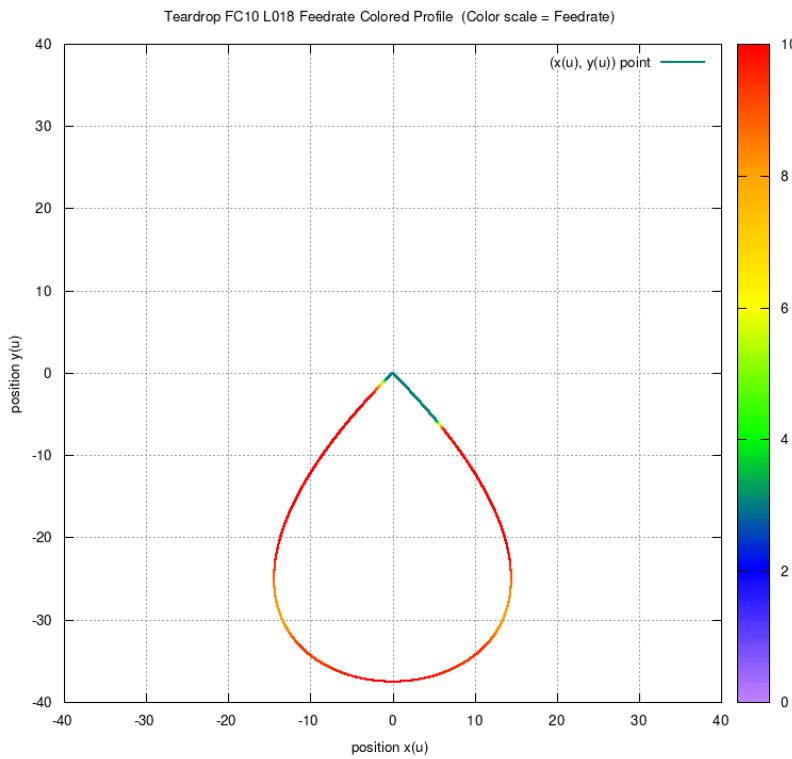


Figure 102: Teardrop FC20 Colored Feedrate Profile data ngcode

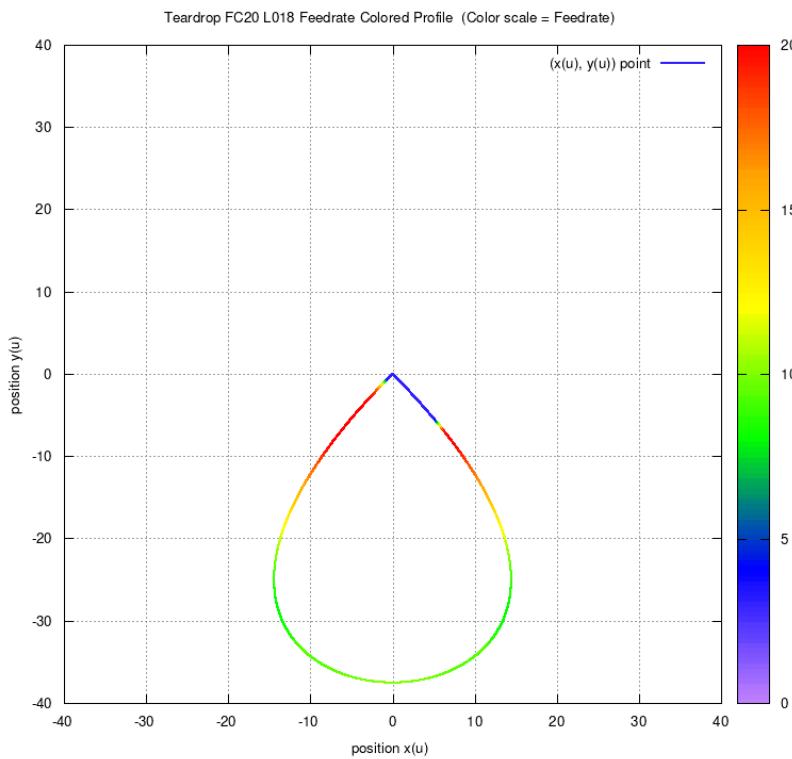


Figure 103: Teardrop FC30 Colored Feedrate Profile data ngcode

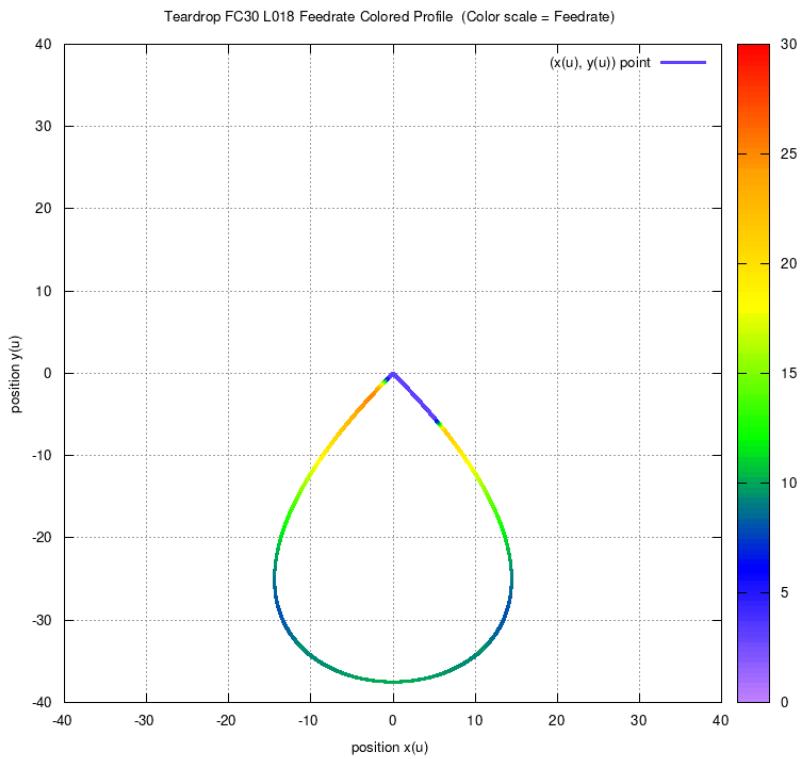


Figure 104: Teardrop FC40 Colored Feedrate Profile data ngcode

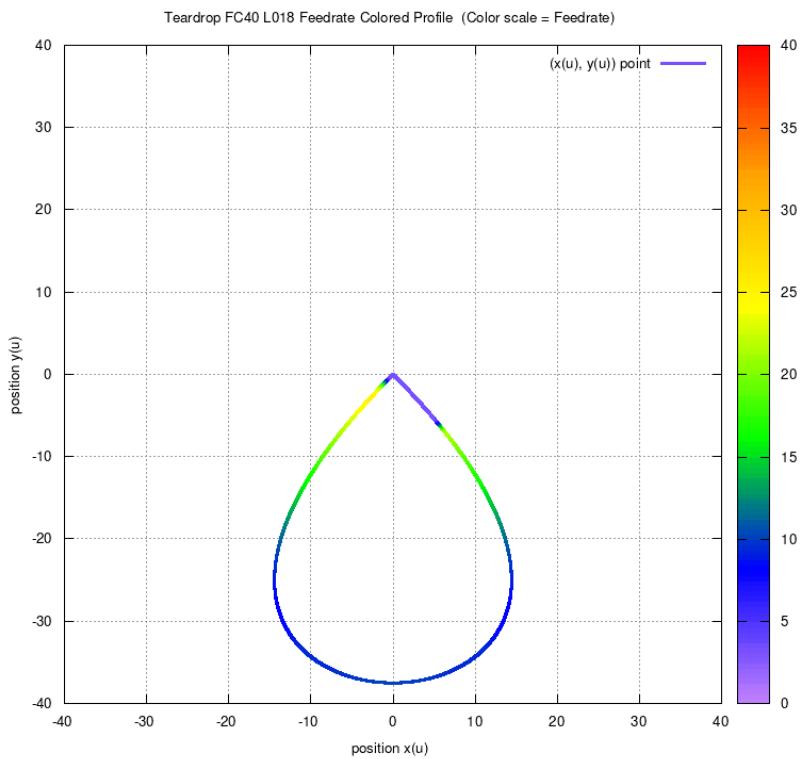


Figure 105: Teardrop FC10 Tangential Acceleration

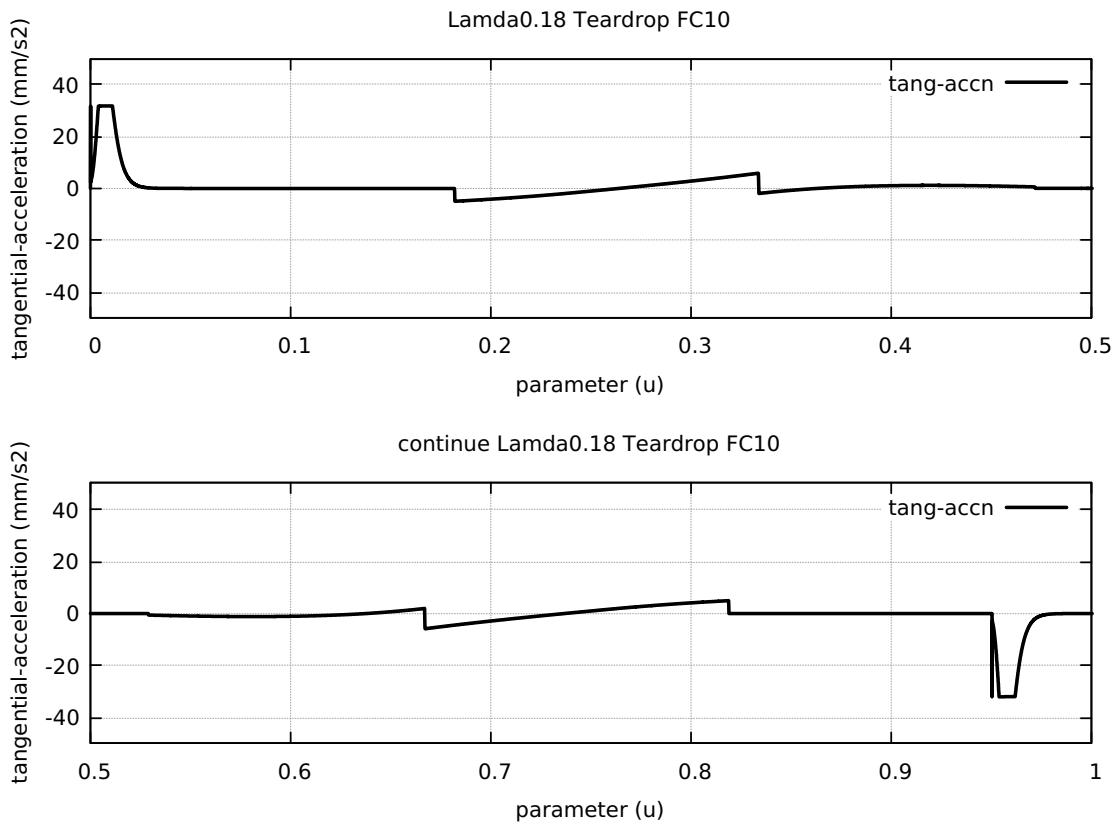


Figure 106: Teardrop FC20 Tangential Acceleration

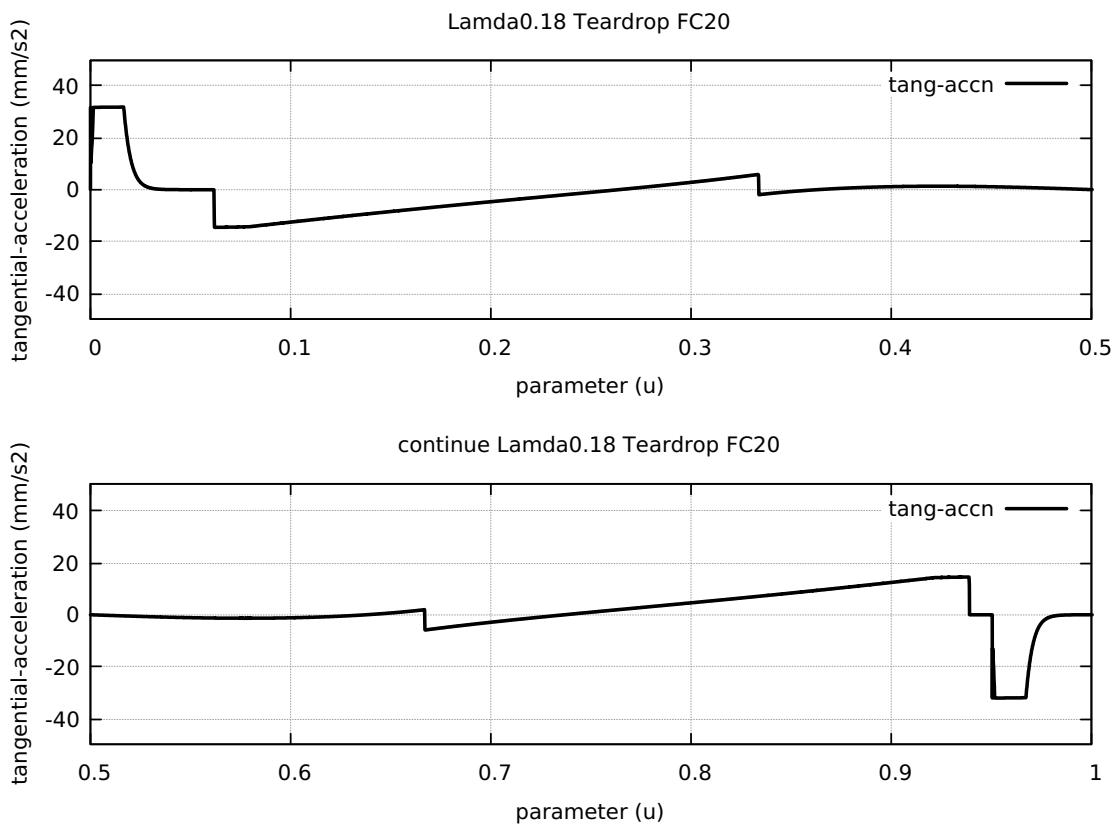


Figure 107: Teardrop FC30 Tangential Acceleration

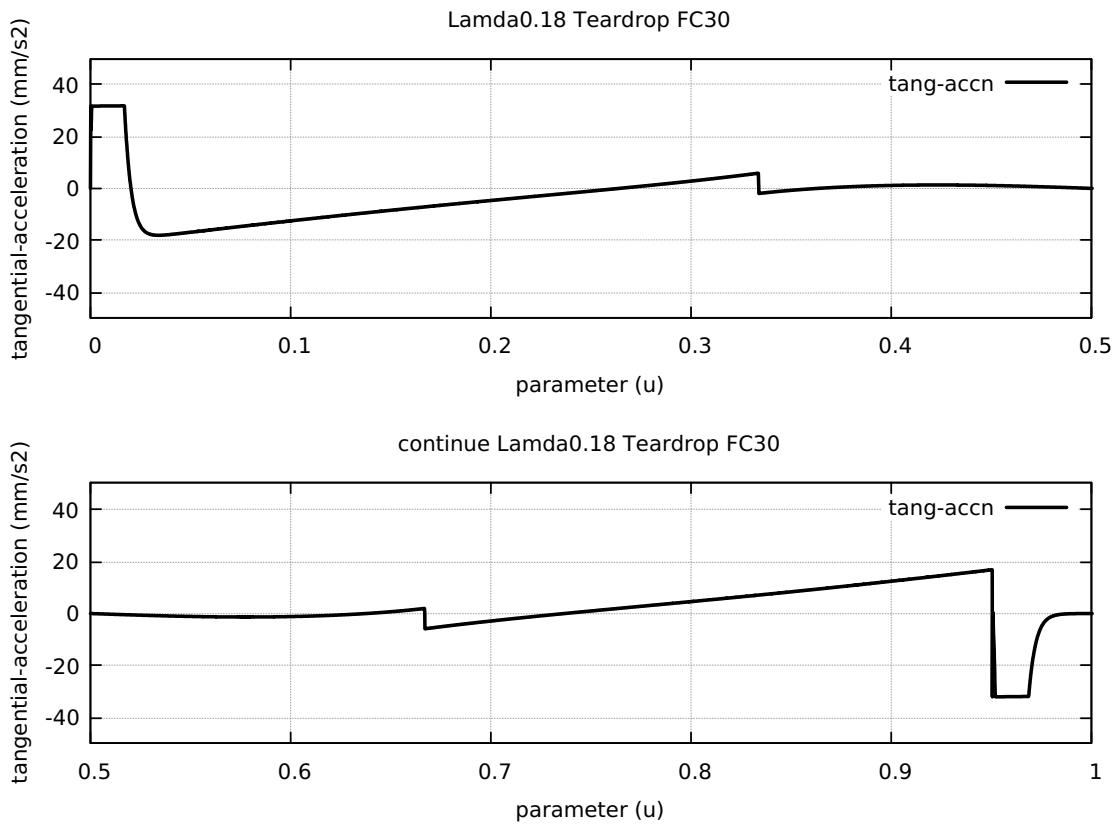


Figure 108: Teardrop FC40 Tangential Acceleration

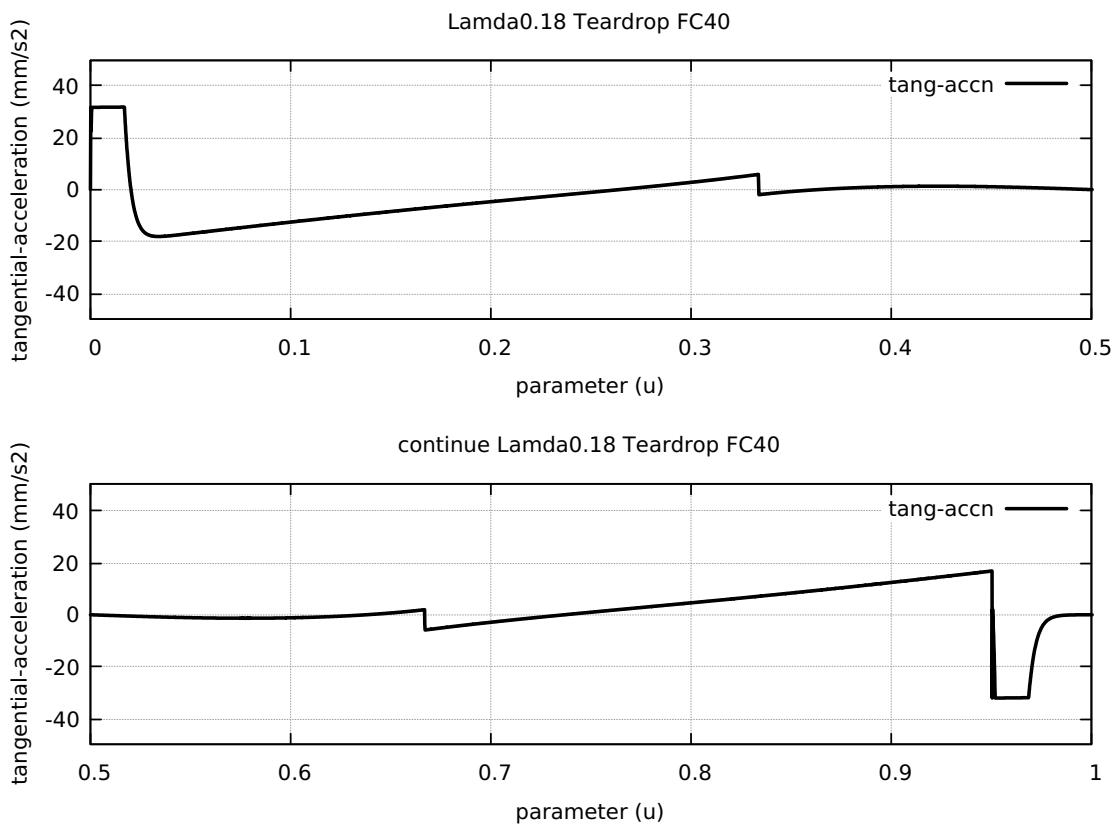


Figure 109: Teardrop FC20 Nominal Separation NAL and NCL

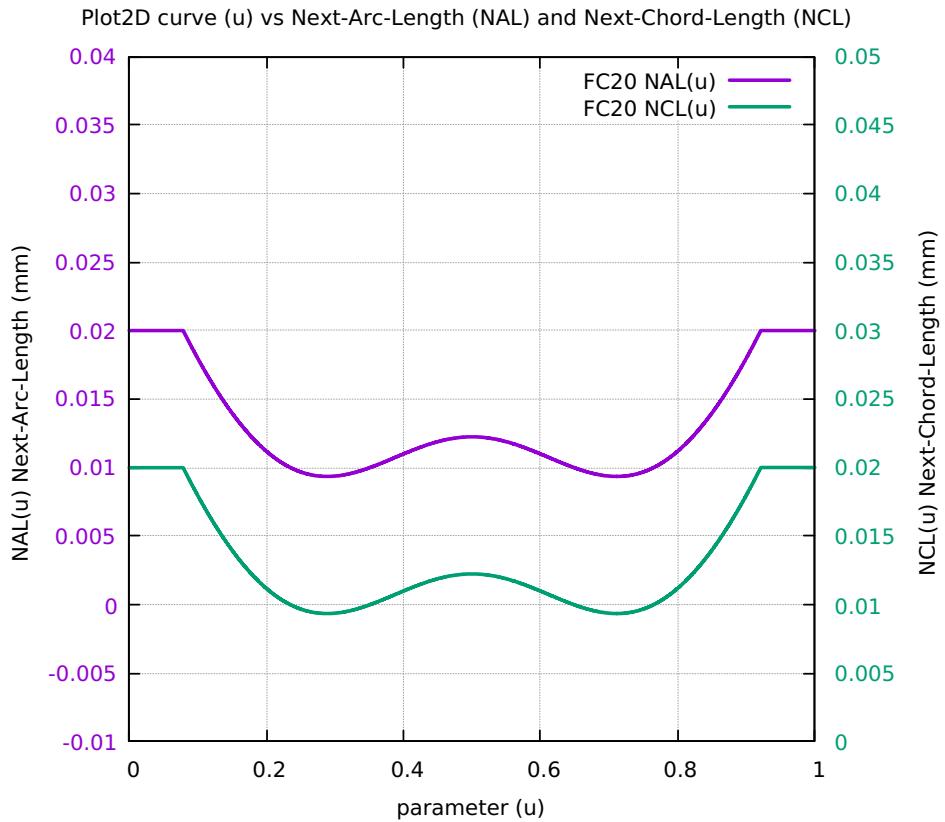


Figure 110: Teardrop Difference SAL minus SCL for FC10 FC20 FC30 FC40

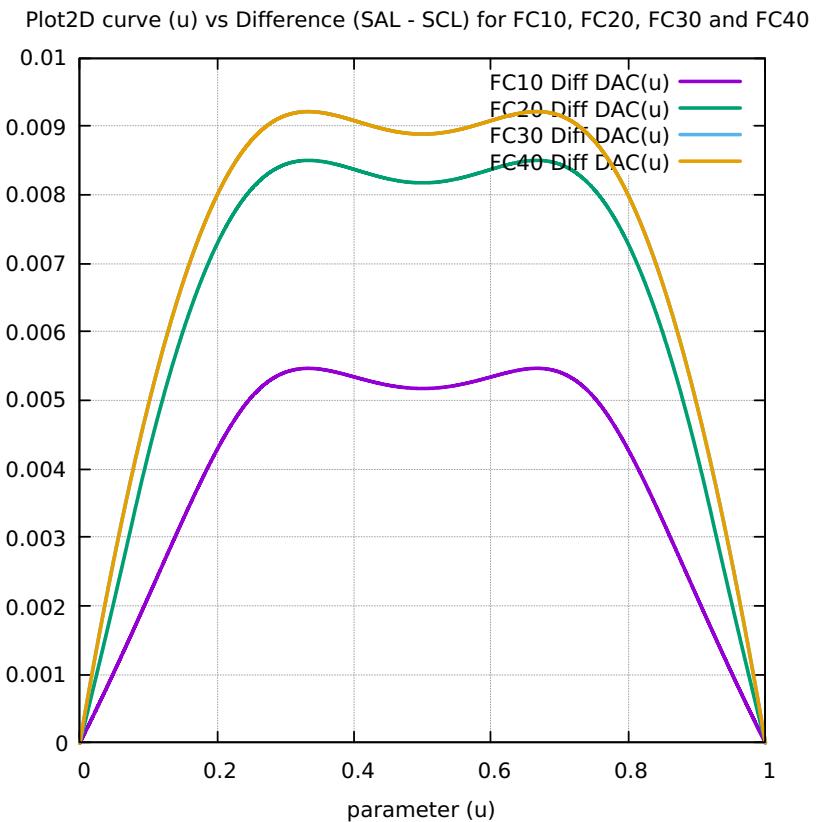


Figure 111: Teardrop FC10 FrateCmd CurrFrate X-Frate Y-Frate

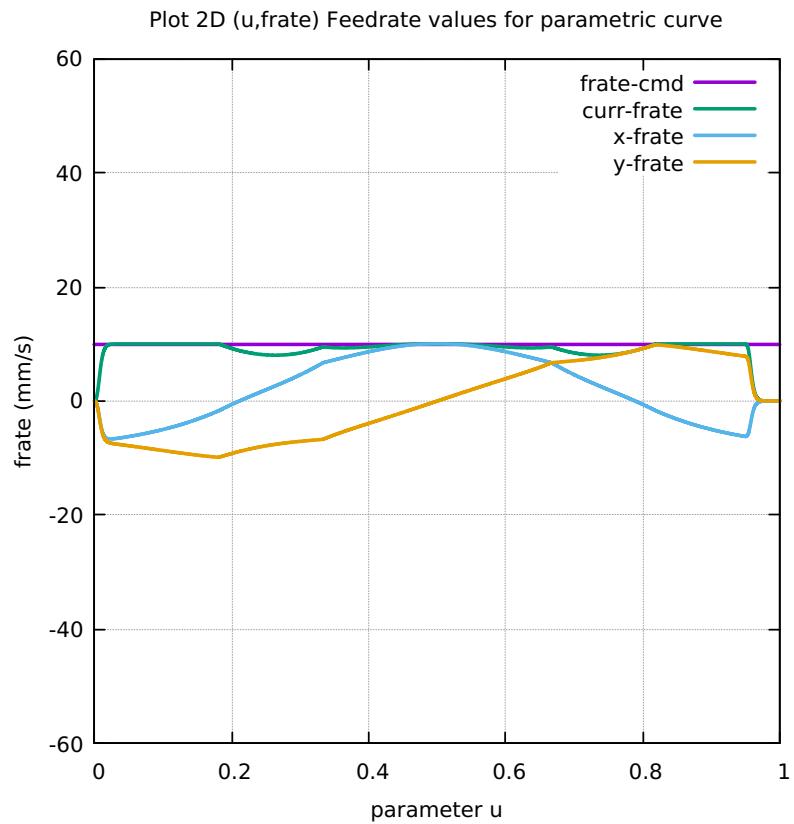


Figure 112: Teardrop FC20 FrateCmd CurrFrate X-Frate Y-Frate

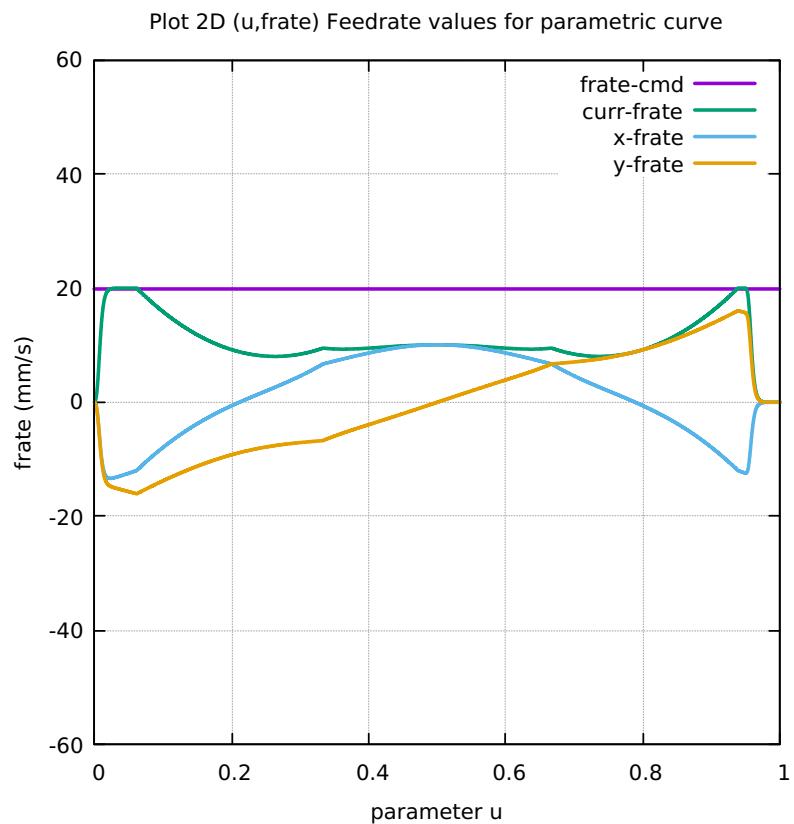


Figure 113: Teardrop FC30 FrateCmd CurrFrate X-Frate Y-Frate

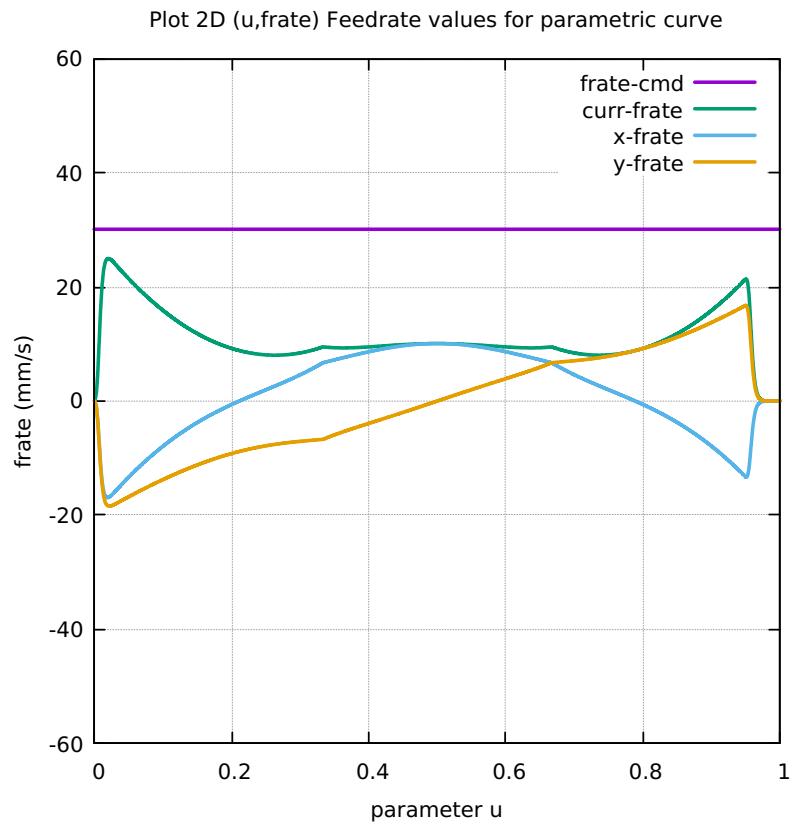


Figure 114: Teardrop FC40 FrateCmd CurrFrate X-Frate Y-Frate

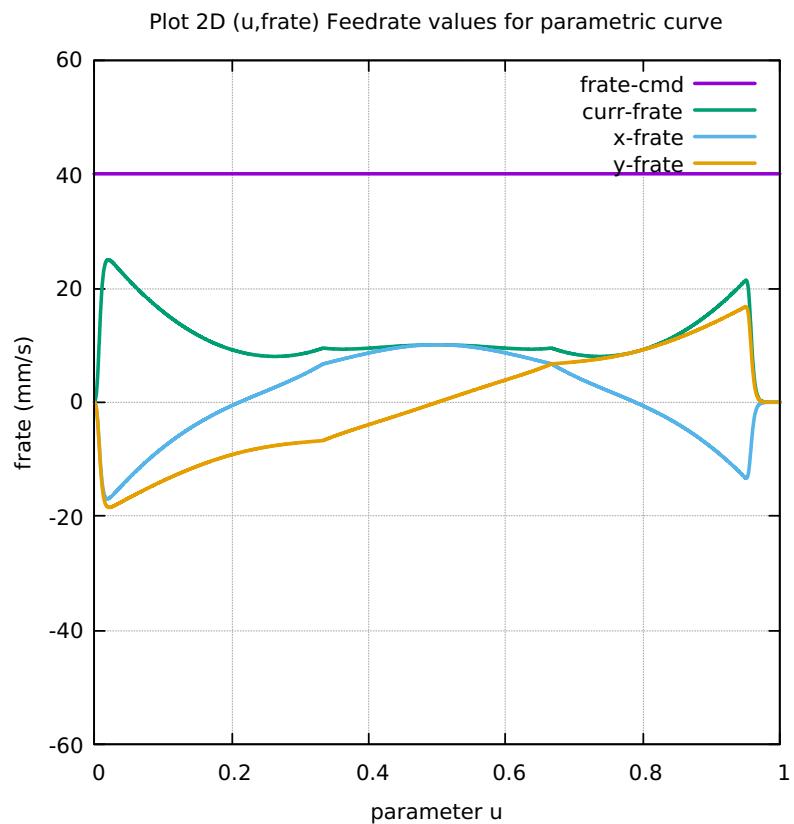


Figure 115: Teardrop FC10 Four Components FeedrateLimit

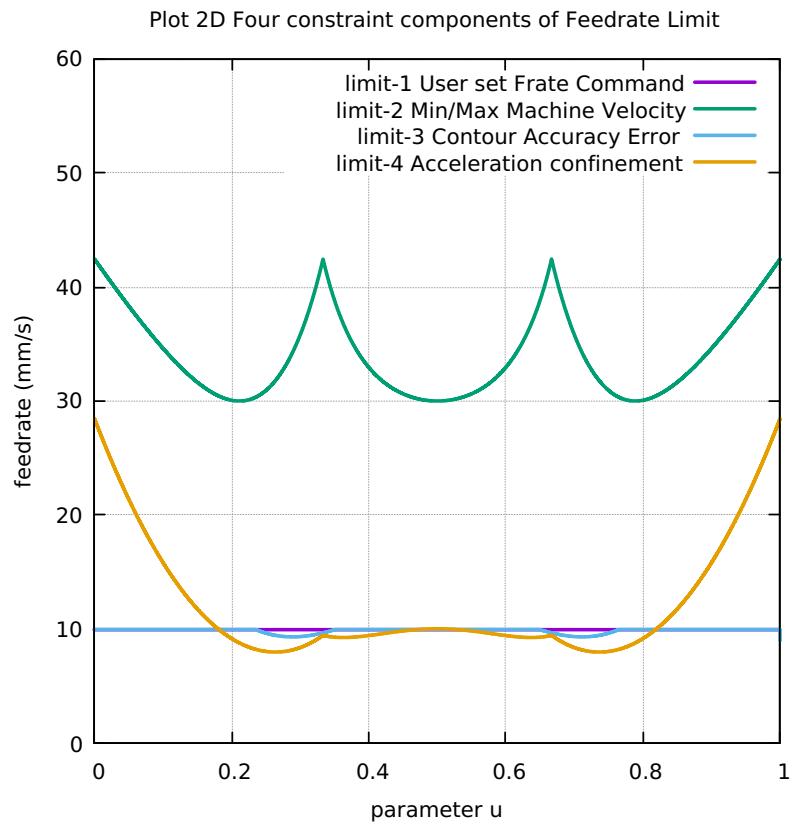


Figure 116: Teardrop FC20 Four Components FeedrateLimit

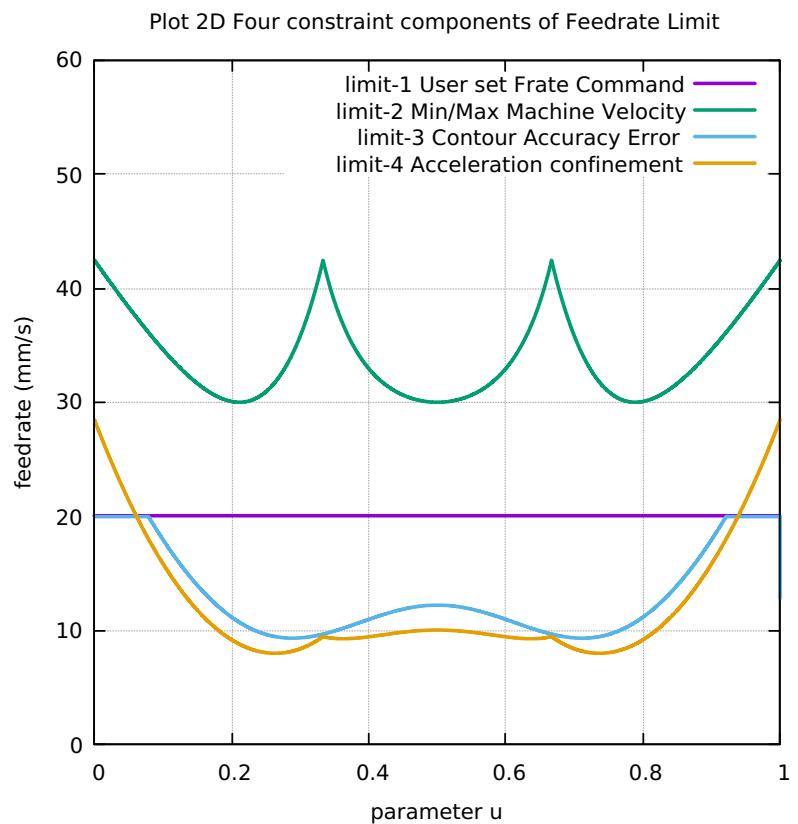


Figure 117: Teardrop FC30 Four Components FeedrateLimit

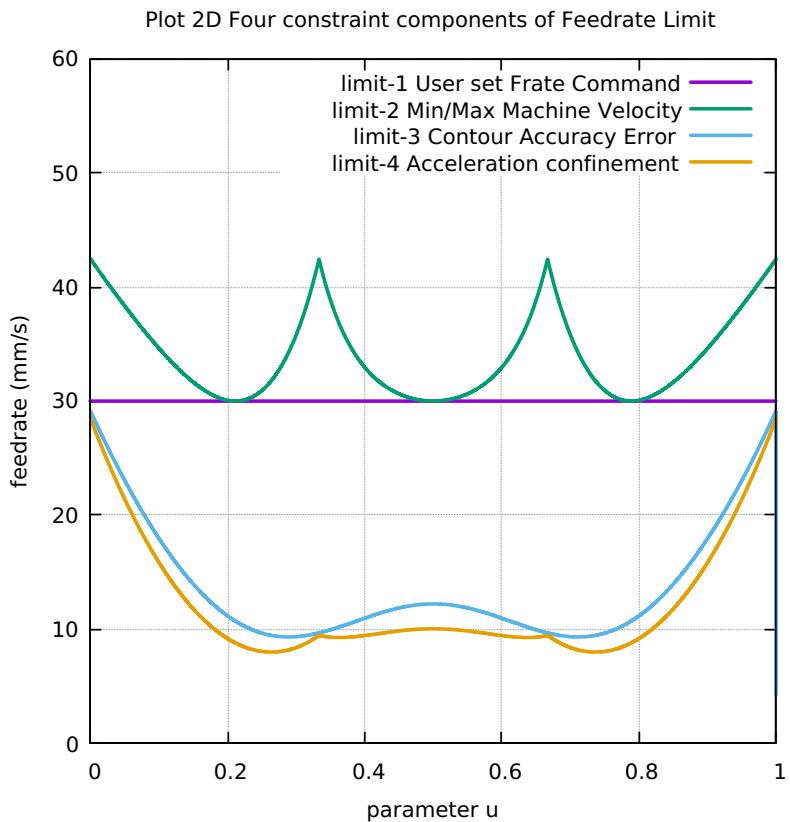


Figure 118: Teardrop FC40 Four Components FeedrateLimit

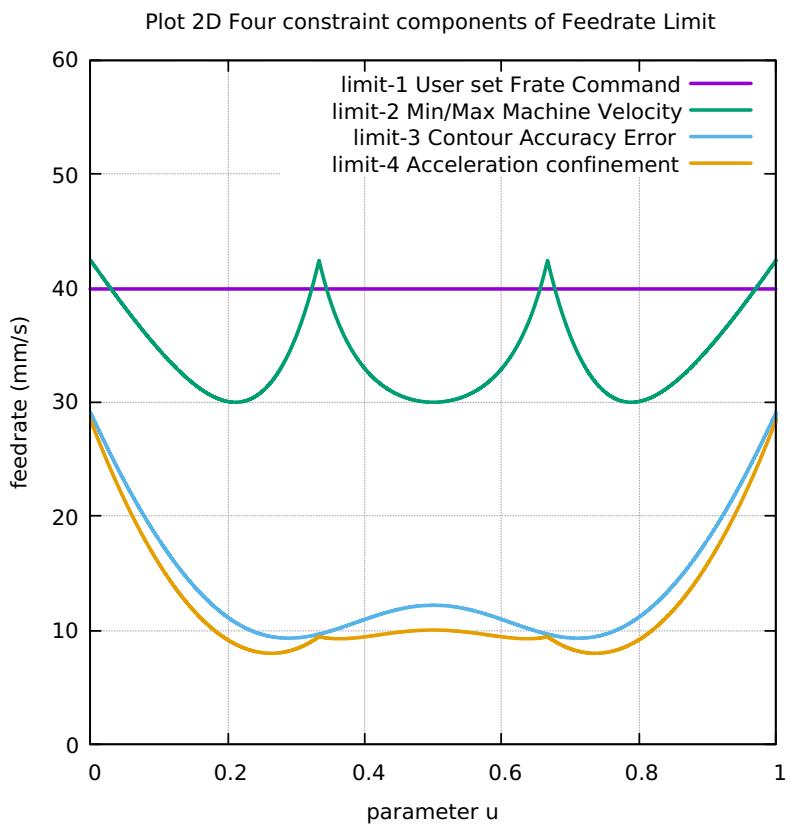


Figure 119: Teardrop Histogram Points FC10 FC20 FC30 FC40

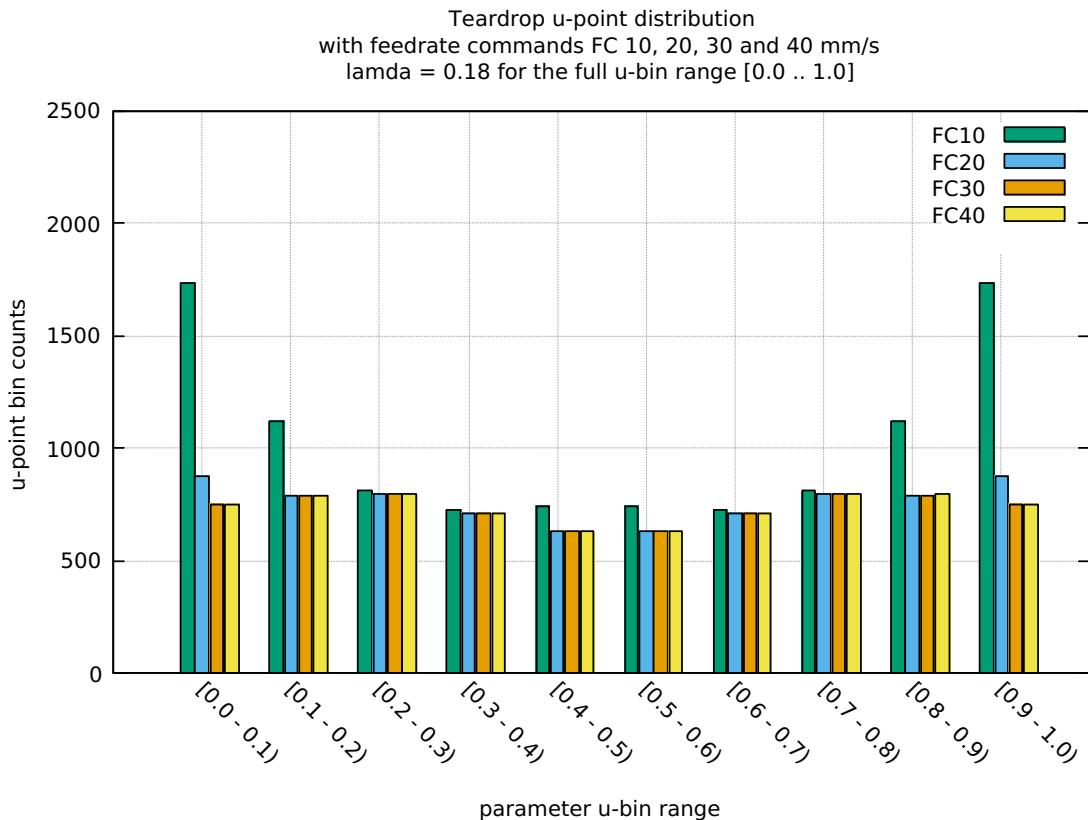


Table 6: Teardrop Table distribution of interpolated points

BINS	FC10	FC20	FC30	FC40
0.0 – 0.1	1734	875	748	748
0.1 – 0.2	1120	791	791	791
0.2 – 0.3	809	794	794	794
0.3 – 0.4	726	710	711	711
0.4 – 0.5	741	629	629	629
0.5 – 0.6	742	629	628	629
0.6 – 0.7	726	710	711	711
0.7 – 0.8	809	794	794	793
0.8 – 0.9	1120	791	791	792
0.9 – 1.0	1734	876	750	749
Total counts	10261	7599	7347	7347

Table 7: Teardrop Table FC10-20-30-40 Run Performance data

1	Curve Type	TEARDROP	TEARDROP	TEARDROP
2	User Feedrate Command FC(mm/s)	FC10	FC20	FC40
3	User Lamda Acceleration Safety Factor	0.18	0.18	0.18
4	Total Interpolated Points (TIP)	10261	7599	7347
5	Total Sum-Chord-Error (SCE) (mm)	5.8097378338076E-03	7.140807162860E-03	7.336793707381E-03
6	Ratio 1 = (SCE/TIP) = Chord-Error/Point	5.662512512745E-07	9.39827212807E-07	9.987467611463E-07
7	Total Sum-Arc-Length (SAL) (mm)	1.018356741269E+02	1.018418663504E+02	1.018595636256E+02
8	Total Sum-Chord-Length (SCL) (mm)	1.018356732173E+02	1.018418655699E+02	1.018595666011E+02
9	Difference = (SAL - SCL) (mm)	9.095903408252E-07	7.805327442156E-07	-2.975420997586E-06
10	Percentage Difference = (SAL - SCL)/SAL	8.931942058845E-07	7.664163788301E-07	-2.921101261067E-06
11	Ratio 2 = (SCE/SCL) = Chord Error/Chord-Length	5.705012452441E-05	7.011661778683E-05	7.202851879506E-05
12	Total Sum-Arc-Theta (SAT) (rad)	4.712304324578E+00	4.712268805770E+00	4.712349787227E+00
13	Total Sum-Arc-Area (SAA) (mm2)	3.822127588087E-05	6.182290957317E-05	6.781012634326E-05
14	Ratio 3 = (SAA/SCL) = Arc-Area/Chord-Length	5.705012452441E-05	7.011661778683E-05	7.202851879506E-05
15	Average-Chord-Error (ACE) (mm)	5.662512512745E-07	9.398272128007E-07	9.987467611463E-07
16	Average-Arc-Length (AAL) (mm)	9.925504300871E-03	1.340377288107E-02	1.386599014779E-02
17	Average-Chord-Length (ACL) (mm)	9.925504212216E-03	1.340377277834E-02	1.386599055283E-02
18	Average-Arc-Theta (AAT) (rad)	4.592889205241E-04	6.201985793327E-04	6.414851330284E-04
19	Average-Arc-Area (AAA) (mm2)	3.725270553691E-09	8.136734610840E-09	9.230891143923E-09
20	Algorithm actual runtime on computer (ART) (s)	4.487434922	19.907344569	28.094412173
				32.96324077

.6 APPENDIX BUTTERFLY CURVE

- .6.1 Plot of Butterfly curve [120]**
- .6.2 Butterfly Radius of Curvature [121]**
- .6.3 Butterfly Validation in LinuxCNC [122]**
- .6.4 Butterfly Direction of Travel 3D [123]**
- .6.5 Butterfly First and Second Order Taylor's Approx [124]**
- .6.6 Butterfly First minus Second Order Taylor's Approx [125]**
- .6.7 Butterfly Separate First Second Order Taylor's Approx [126]**
- .6.8 Butterfly Separation SAL and SCL [127]**
- .6.9 Butterfly Chord-error in close view 2 scales [128]**
- .6.10 Butterfly Four Components Feedrate Limit [129]**
- .6.11 Butterfly FrateCommand FrateLimit and Curr-Frate [130]**
- .6.12 Butterfly FeedRateLimit minus CurrFeedRate [131]**
- .6.13 Butterfly FC20-Nominal X and Y Feedrate Profiles [132]**
- .6.14 Butterfly FC20 Nominal Tangential Acceleration [133]**
- .6.15 Butterfly FC20 Nominal Rising S-Curve Profile [134]**
- .6.16 Butterfly FC20 Nominal Falling S-Curve Profile [135]**
- .6.17 Butterfly FC10 Colored Feedrate Profile data ngcode [136]**
- .6.18 Butterfly FC20 Colored Feedrate Profile data ngcode [137]**

- .6.19 **Butterfly FC30 Colored Feedrate Profile data ngcode [138]**
- .6.20 **Butterfly FC40 Colored Feedrate Profile data ngcode [139]**
- .6.21 **Butterfly FC10 Tangential Acceleration [140]**
- .6.22 **Butterfly FC20 Tangential Acceleration [141]**
- .6.23 **Butterfly FC30 Tangential Acceleration [142]**
- .6.24 **Butterfly FC40 Tangential Acceleration [143]**
- .6.25 **Butterfly FC20 Nominal Separation NAL and NCL [144]**
- .6.26 **Butterfly SAL minus SCL for FC10 FC20 FC30 FC40 [145]**
- .6.27 **Butterfly FC10 FrateCmd CurrFrate X-Frate Y-Frate [146]**
- .6.28 **Butterfly FC20 FrateCmd CurrFrate X-Frate Y-Frate [147]**
- .6.29 **Butterfly FC30 FrateCmd CurrFrate X-Frate Y-Frate [148]**
- .6.30 **Butterfly FC40 FrateCmd CurrFrate X-Frate Y-Frate [149]**
- .6.31 **Butterfly FC10 Four Components FeedrateLimit [150]**
- .6.32 **Butterfly FC20 Four Components FeedrateLimit [151]**
- .6.33 **Butterfly FC30 Four Components FeedrateLimit [152]**
- .6.34 **Butterfly FC40 Four Components FeedrateLimit [153]**
- .6.35 **Butterfly Histogram Points FC10 FC20 FC30 FC40 [154]**
- .6.36 **Butterfly Table distribution of interpolated points [8]**
- .6.37 **Butterfly Table FC10-20-30-40 Run Performance data [9]**

Figure 120: Plot of Butterfly curve

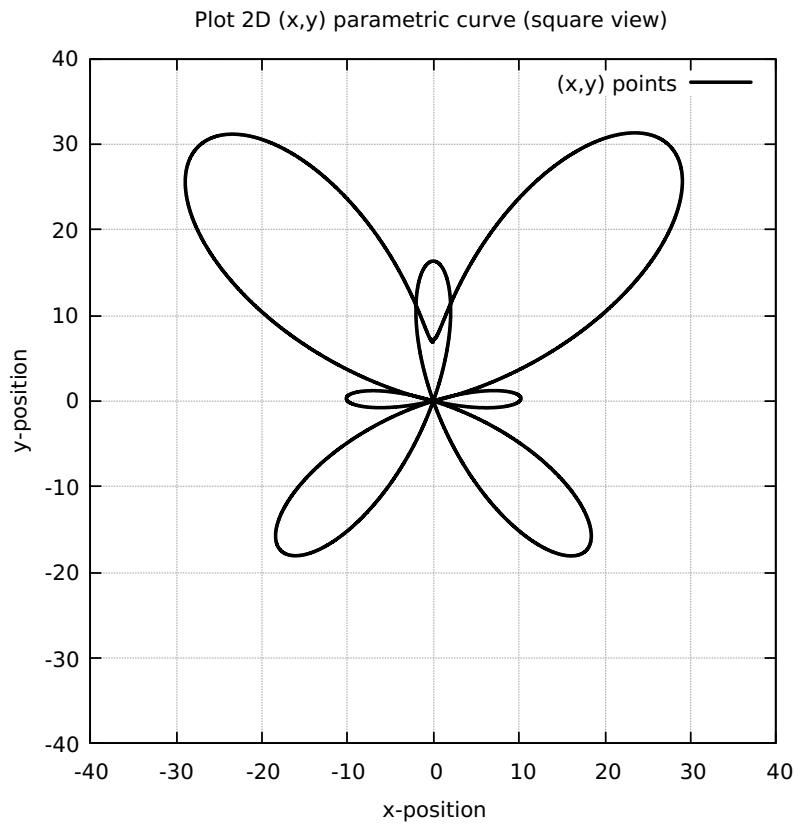


Figure 121: Butterfly Radius of Curvature

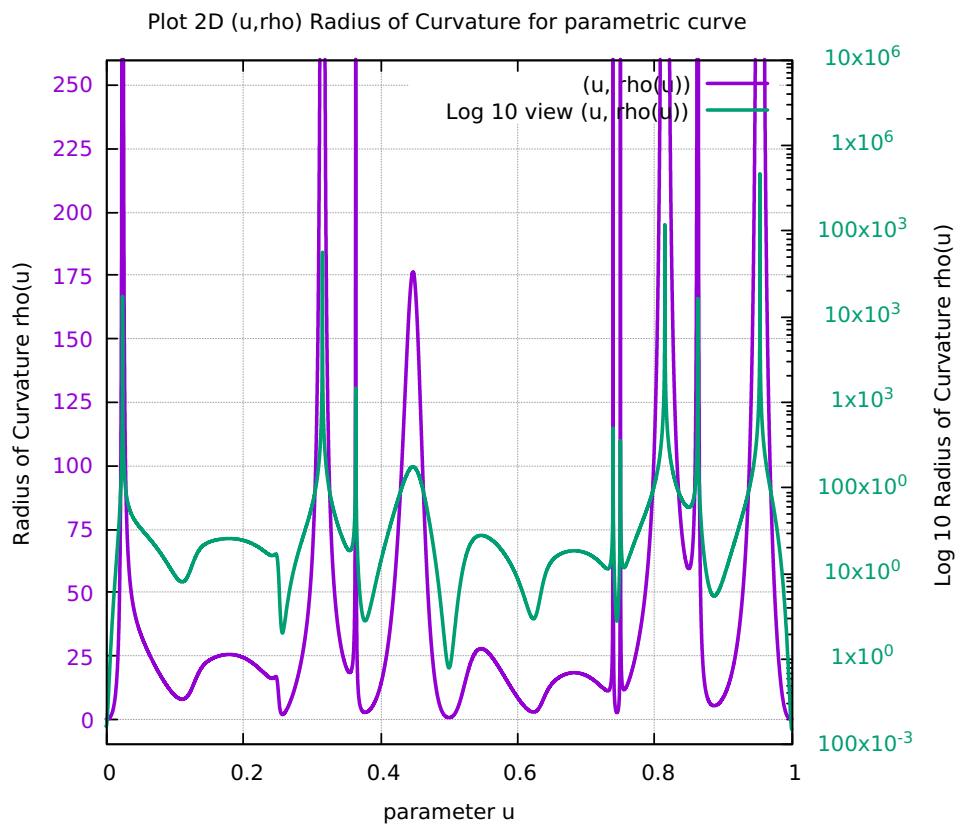


Figure 122: Butterfly Validation in LinuxCNC

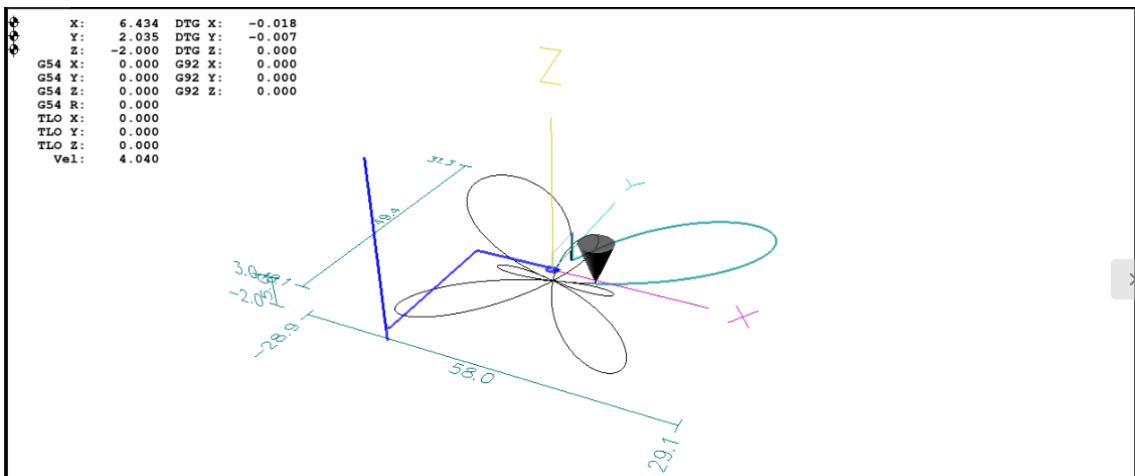


Figure 123: Butterfly Direction of Travel 3D

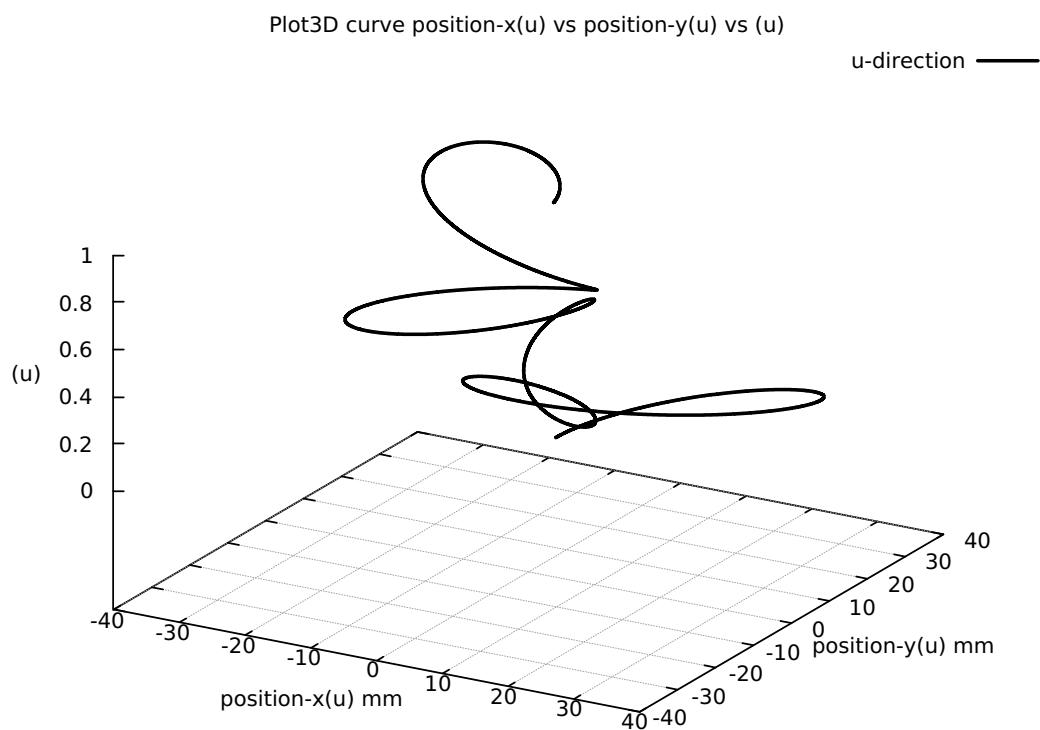


Figure 124: Butterfly First and Second Order Taylor's Approximation

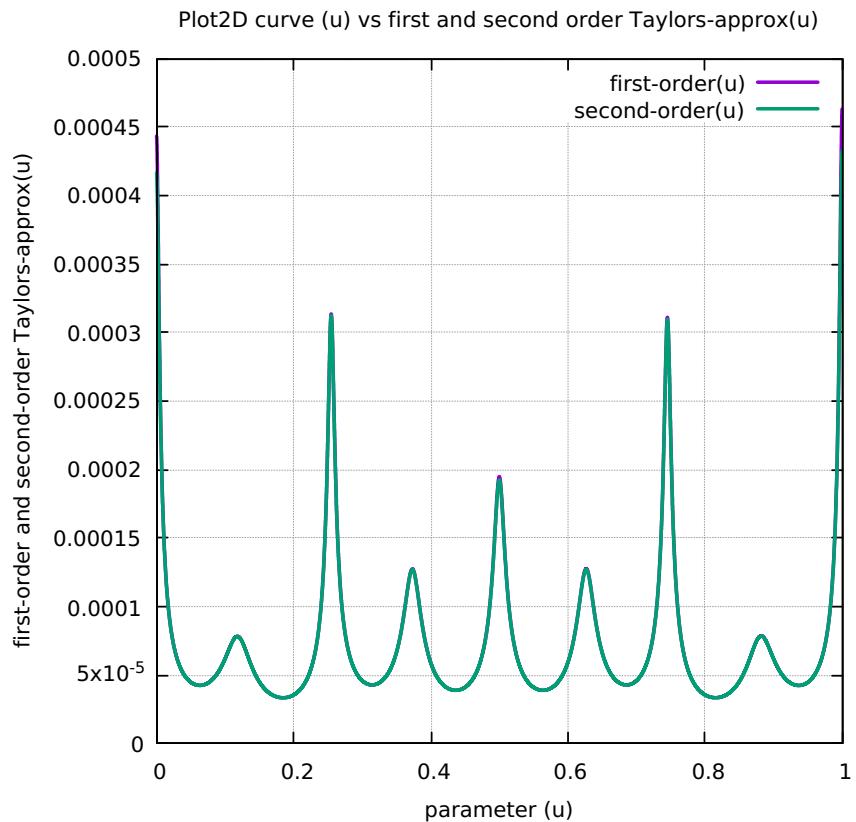


Figure 125: Butterfly First minus Second Order Taylor's Approximation

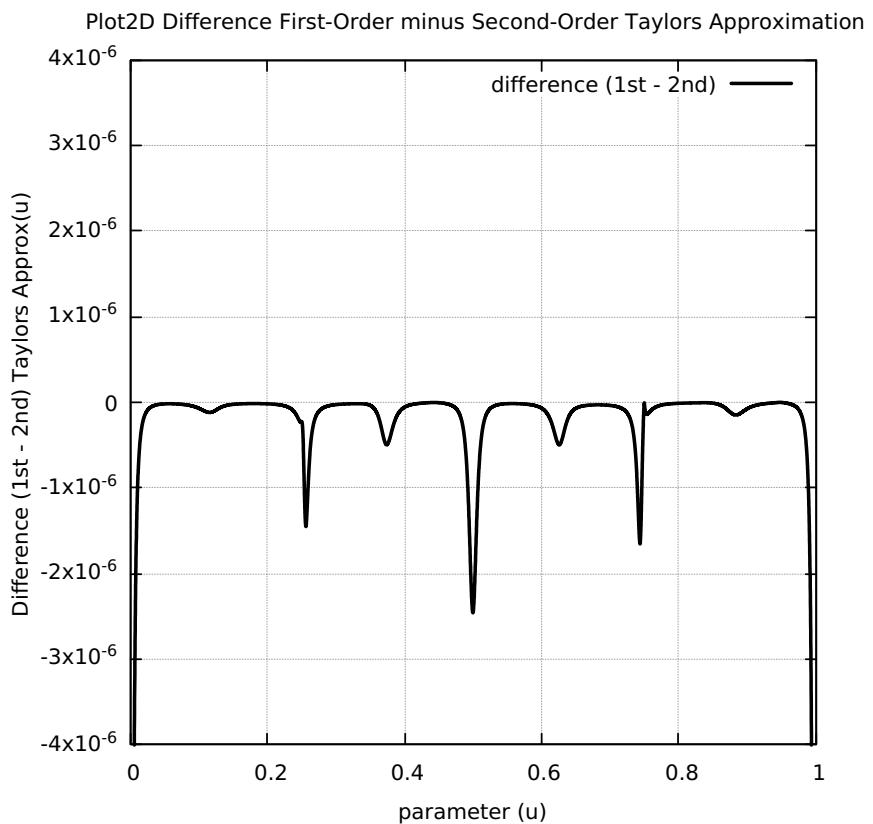


Figure 126: Butterfly Separation First and Second Order Taylor's Approximation

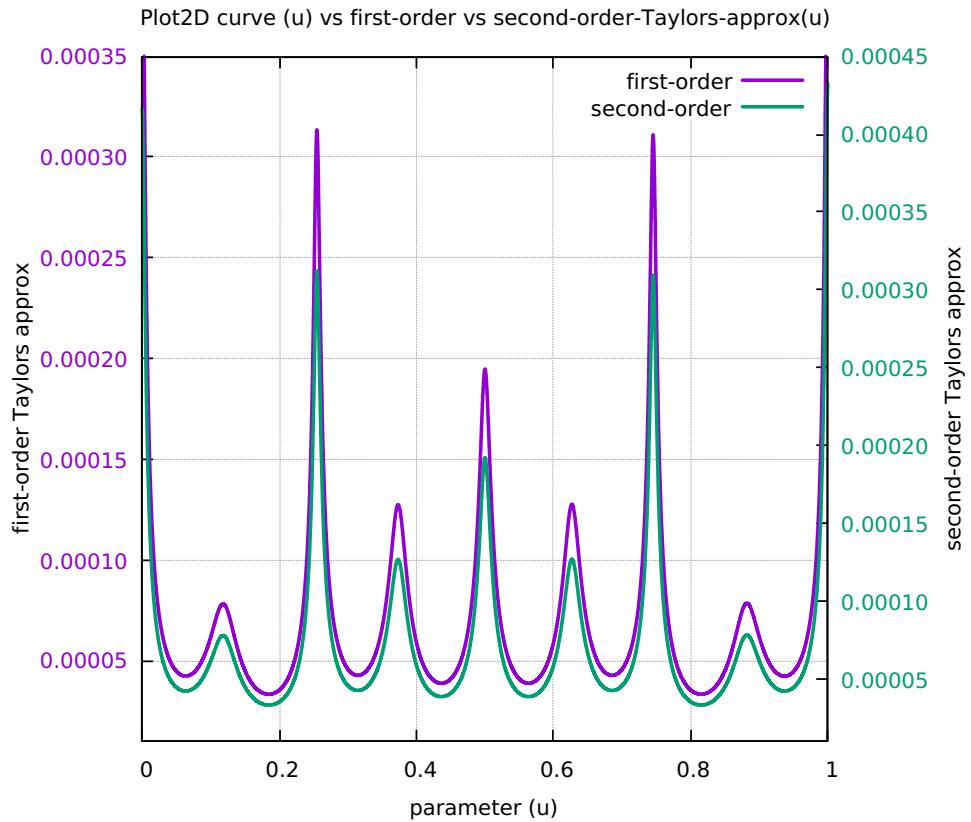


Figure 127: Butterfly Separation SAL and SCL

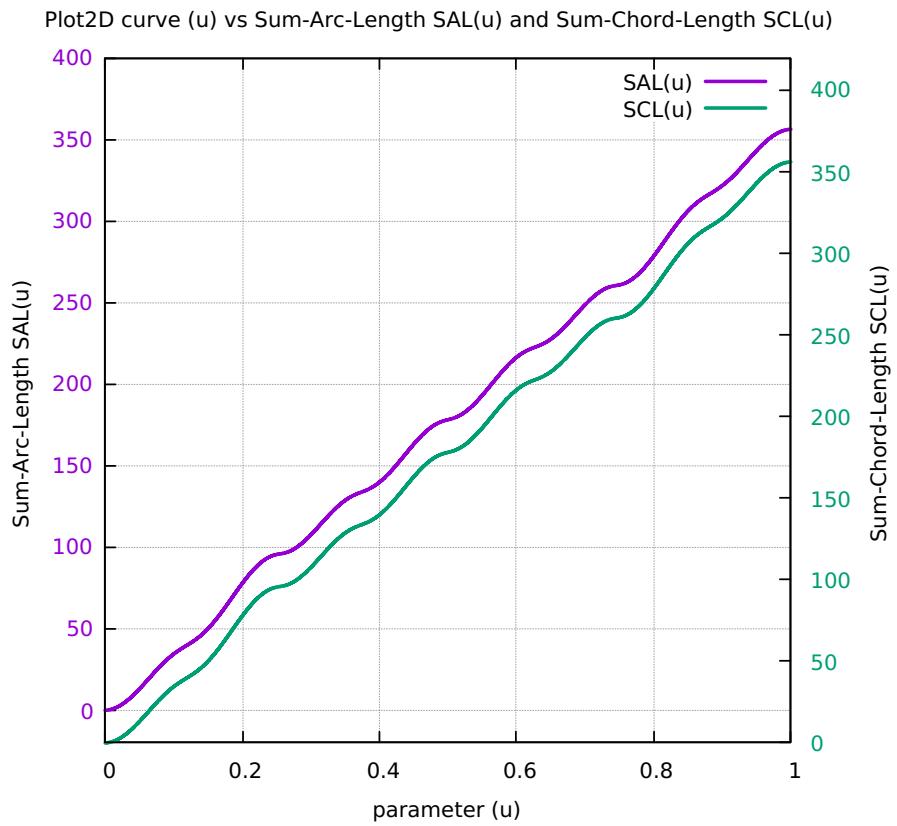


Figure 128: Butterfly Chord-error in close view 2 scales

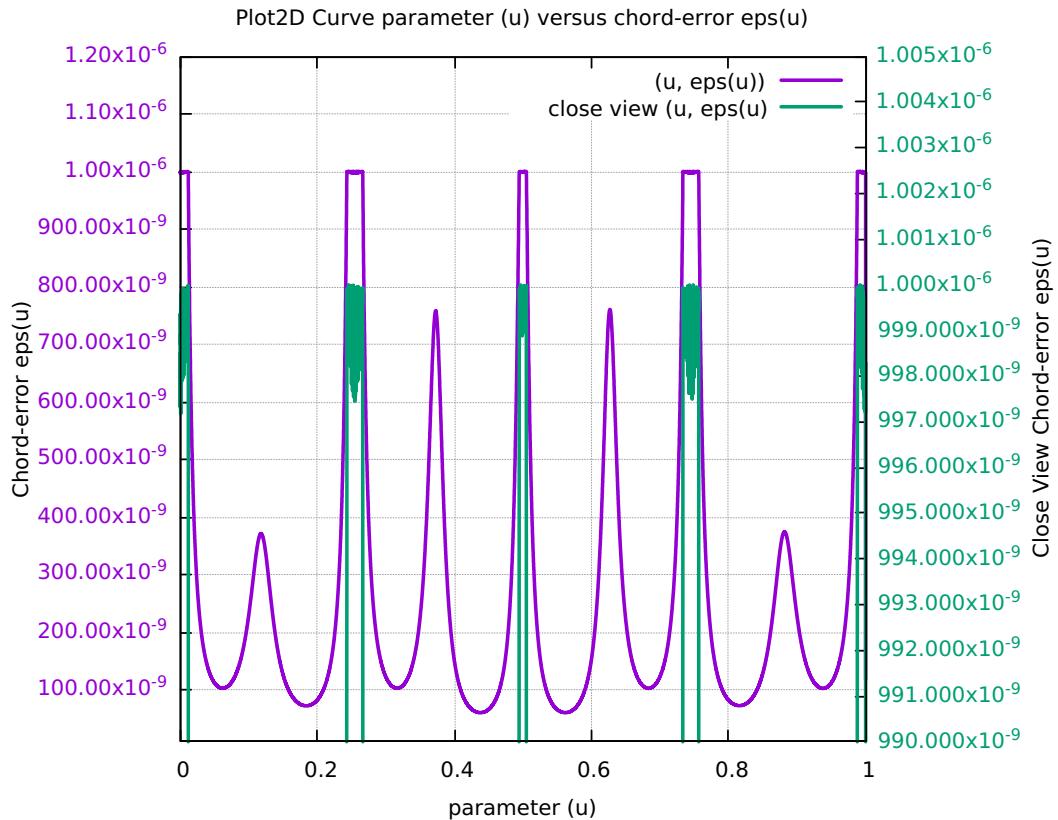


Figure 129: Butterfly Four Components Feedrate Limit

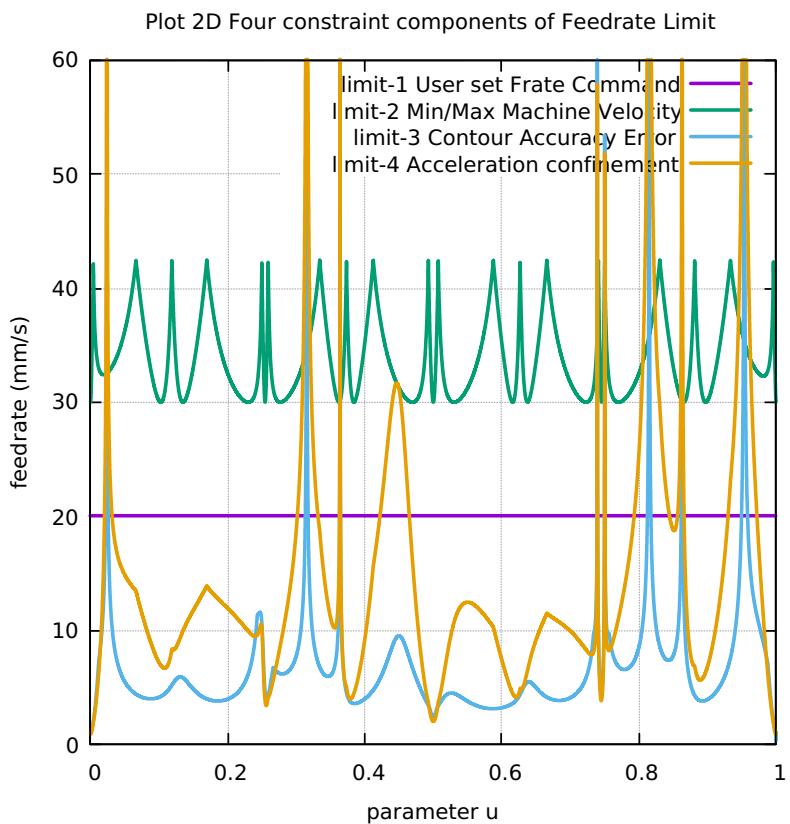


Figure 130: Butterfly FcateCommand FcateLimit and Curr-Fcate

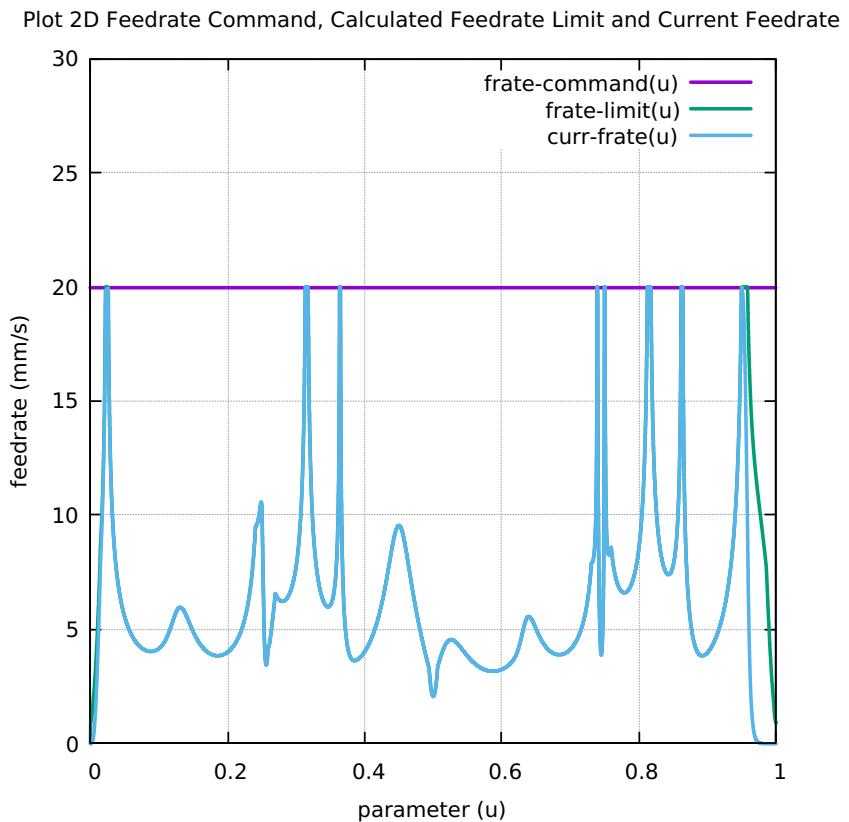


Figure 131: Butterfly FeedRateLimit minus CurrFeedRate

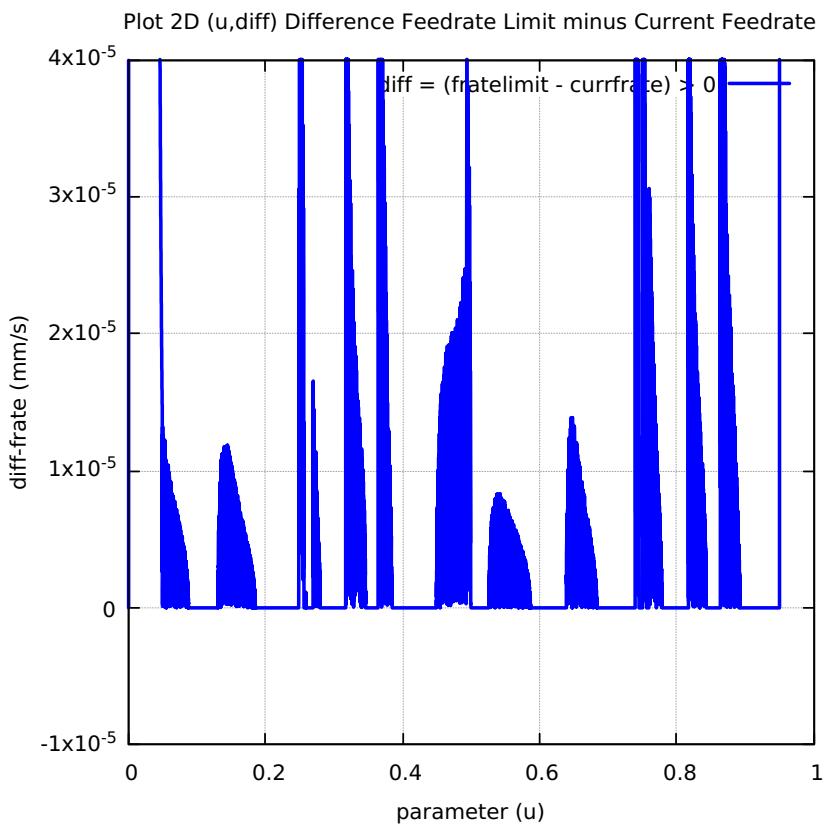


Figure 132: Butterfly FC20-Nominal X and Y Feedrate Profiles

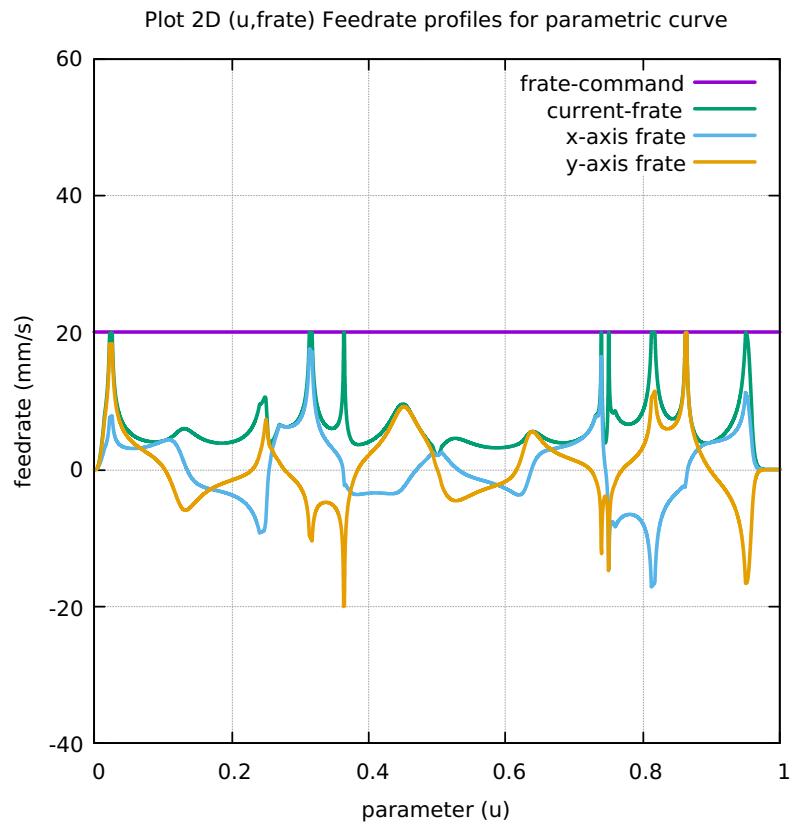


Figure 133: Butterfly FC20 Nominal Tangential Acceleration

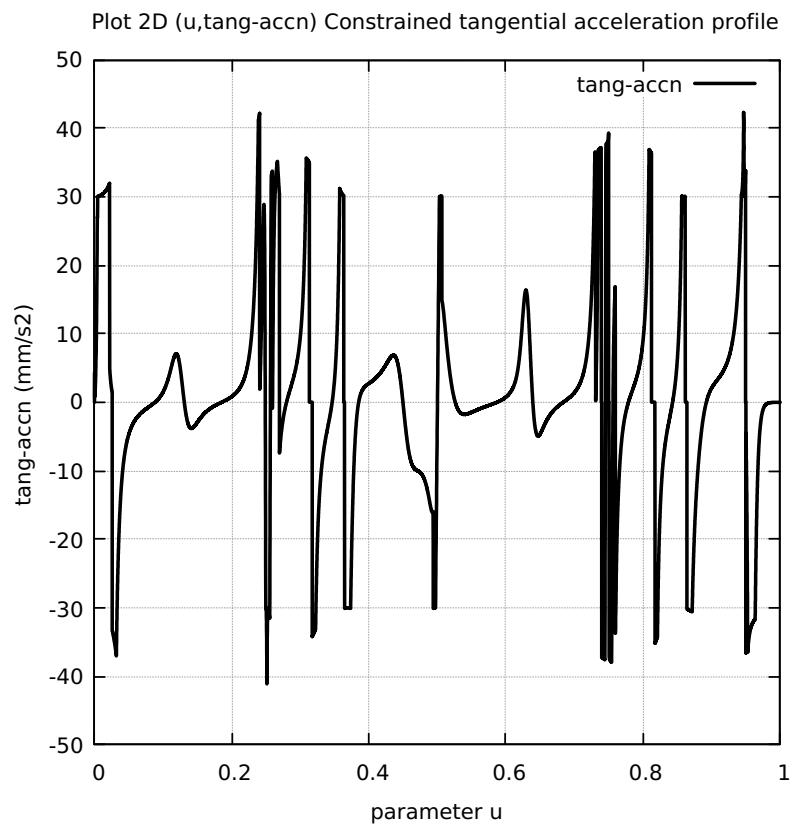


Figure 134: Butterfly FC20 Nominal Rising S-Curve Profile

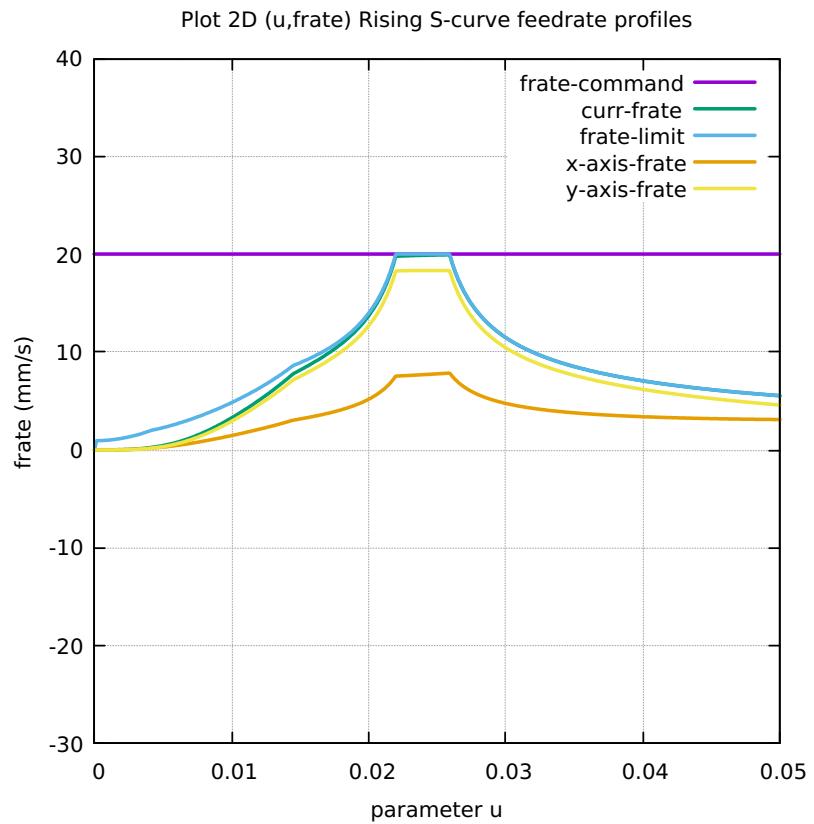


Figure 135: Butterfly FC20 Nominal Falling S-Curve Profile

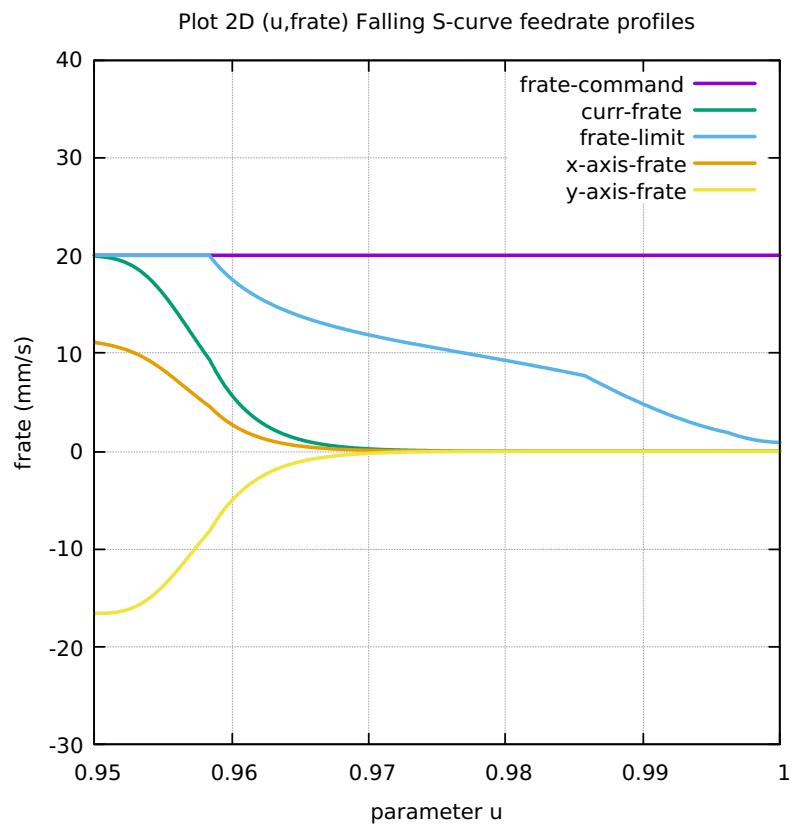


Figure 136: Butterfly FC10 Colored Feedrate Profile data ngcode

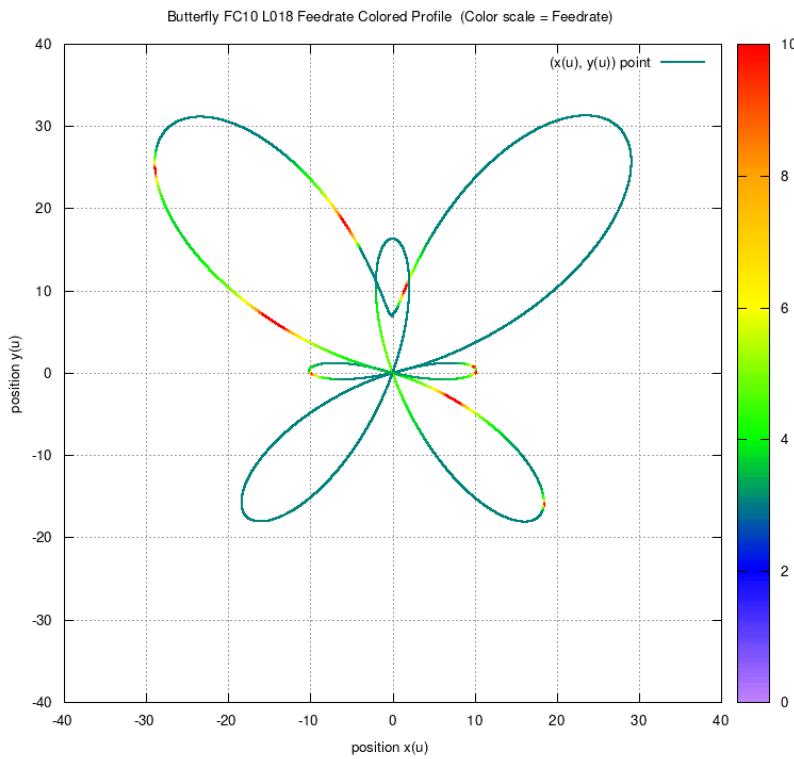


Figure 137: Butterfly FC20 Colored Feedrate Profile data ngcode

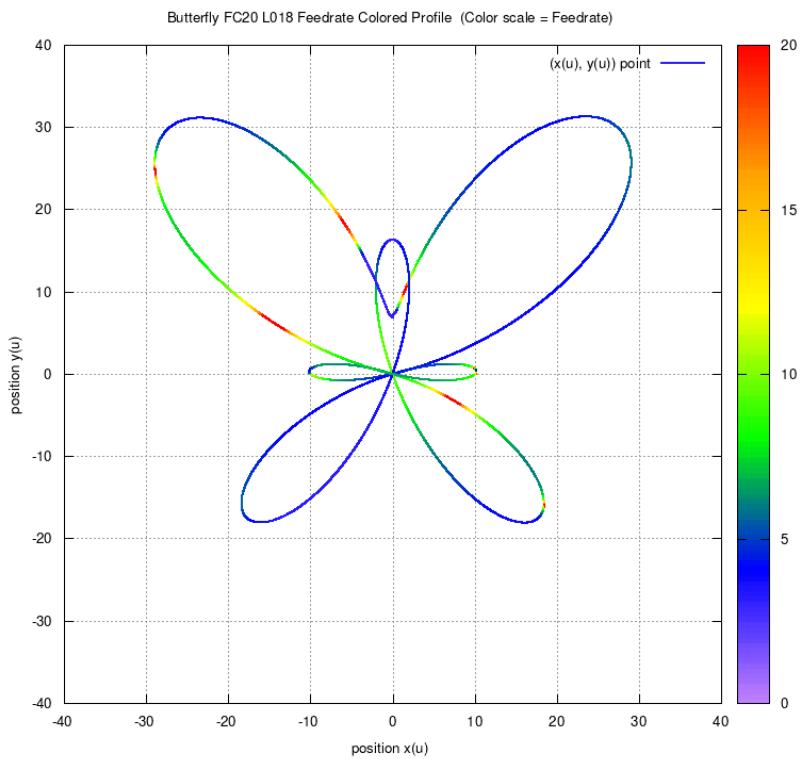


Figure 138: Butterfly FC30 Colored Feedrate Profile data ngcode

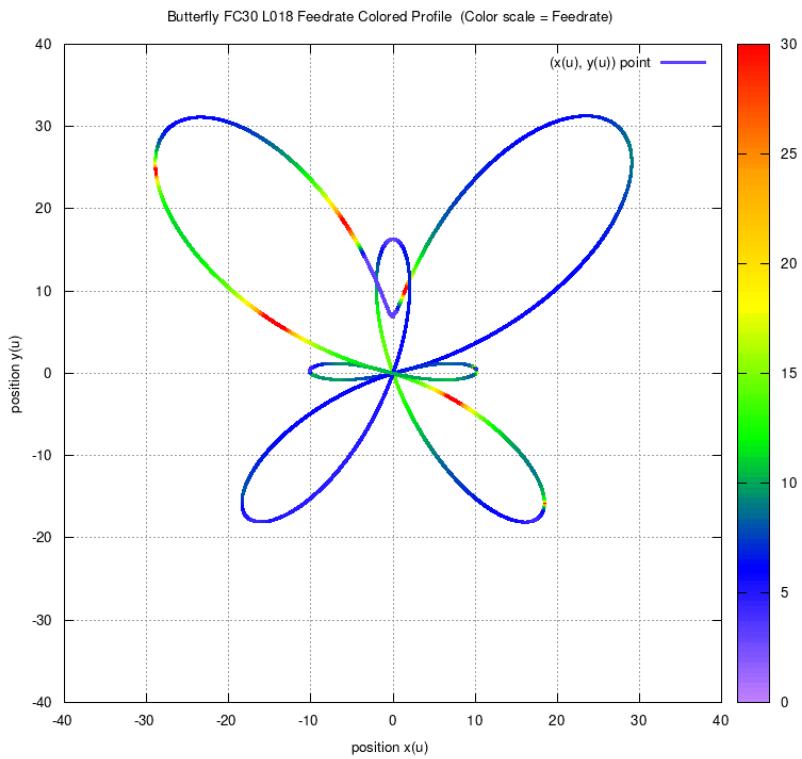


Figure 139: Butterfly FC40 Colored Feedrate Profile data ngcode

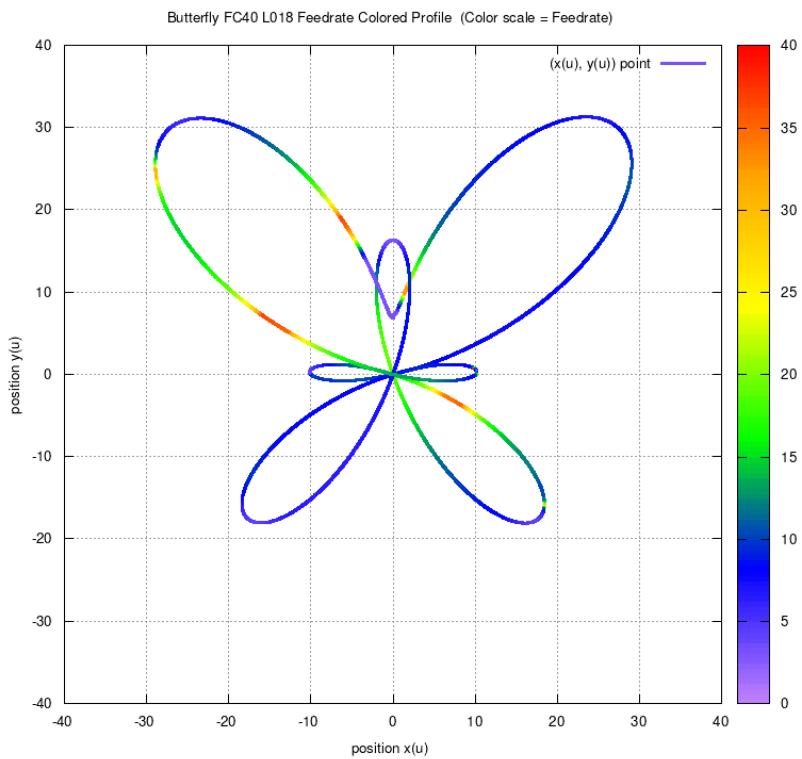


Figure 140: Butterfly FC10 Tangential Acceleration

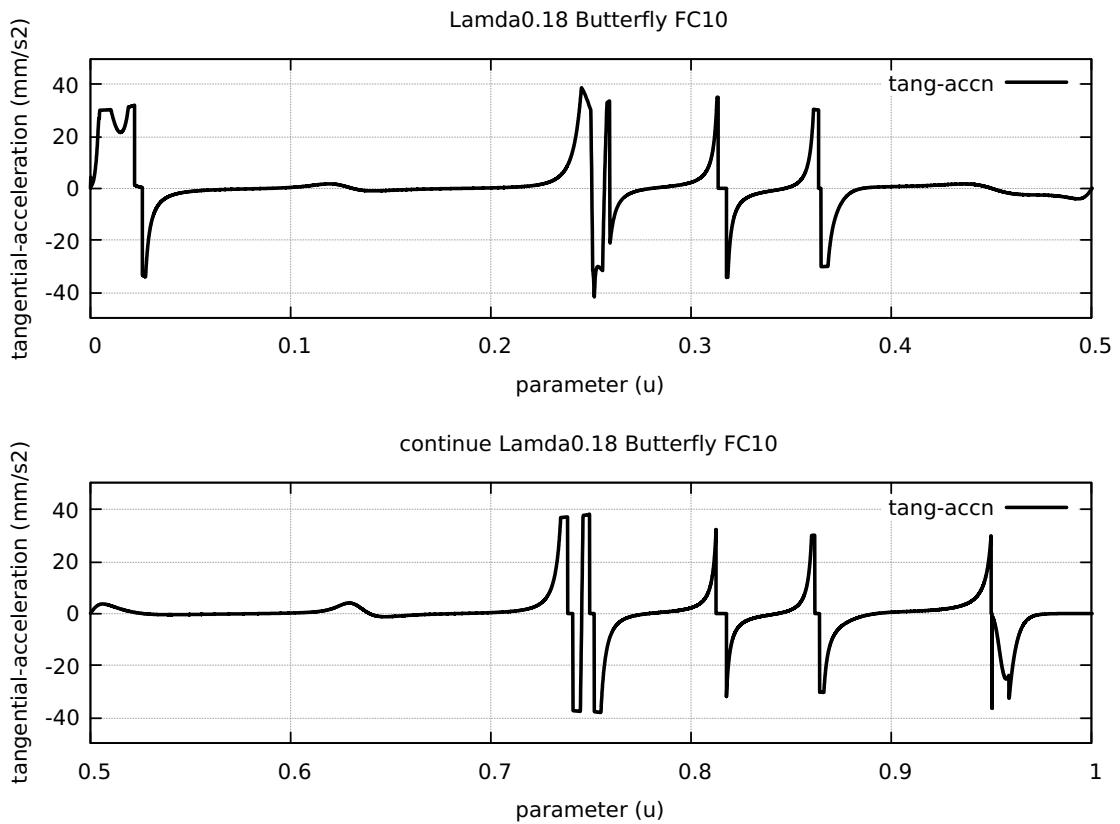


Figure 141: Butterfly FC20 Tangential Acceleration

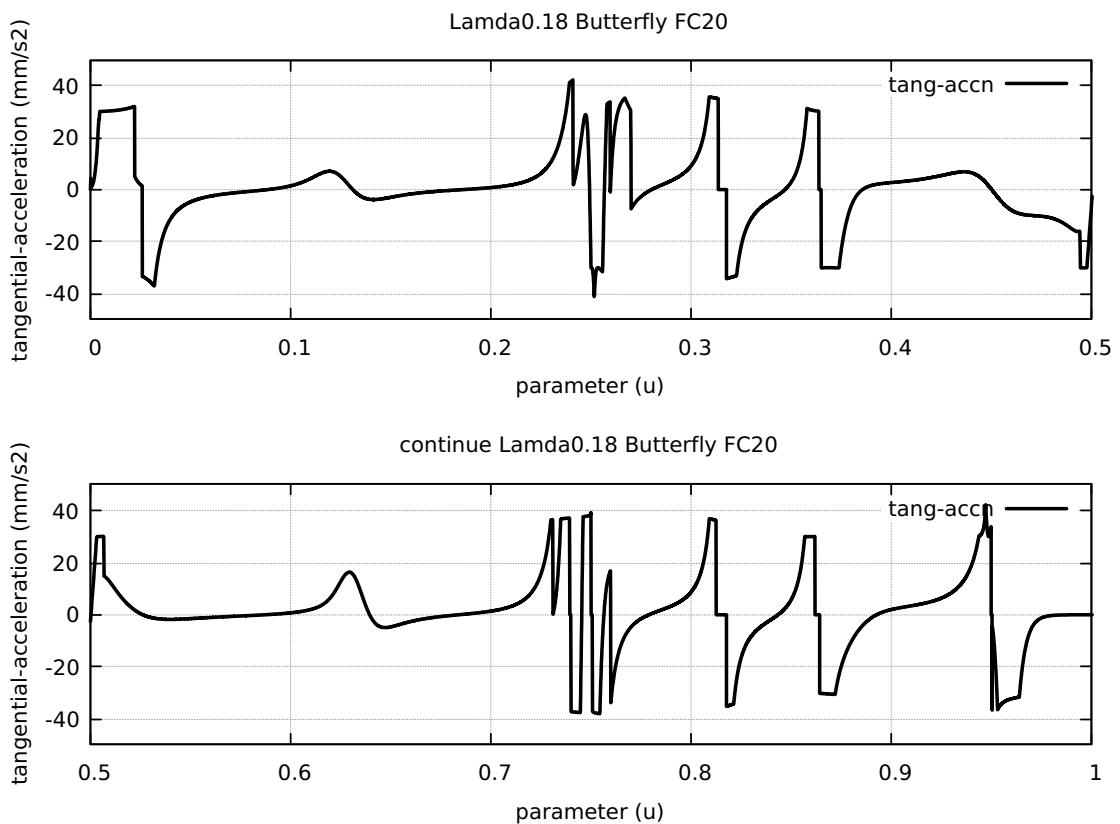


Figure 142: Butterfly FC30 Tangential Acceleration

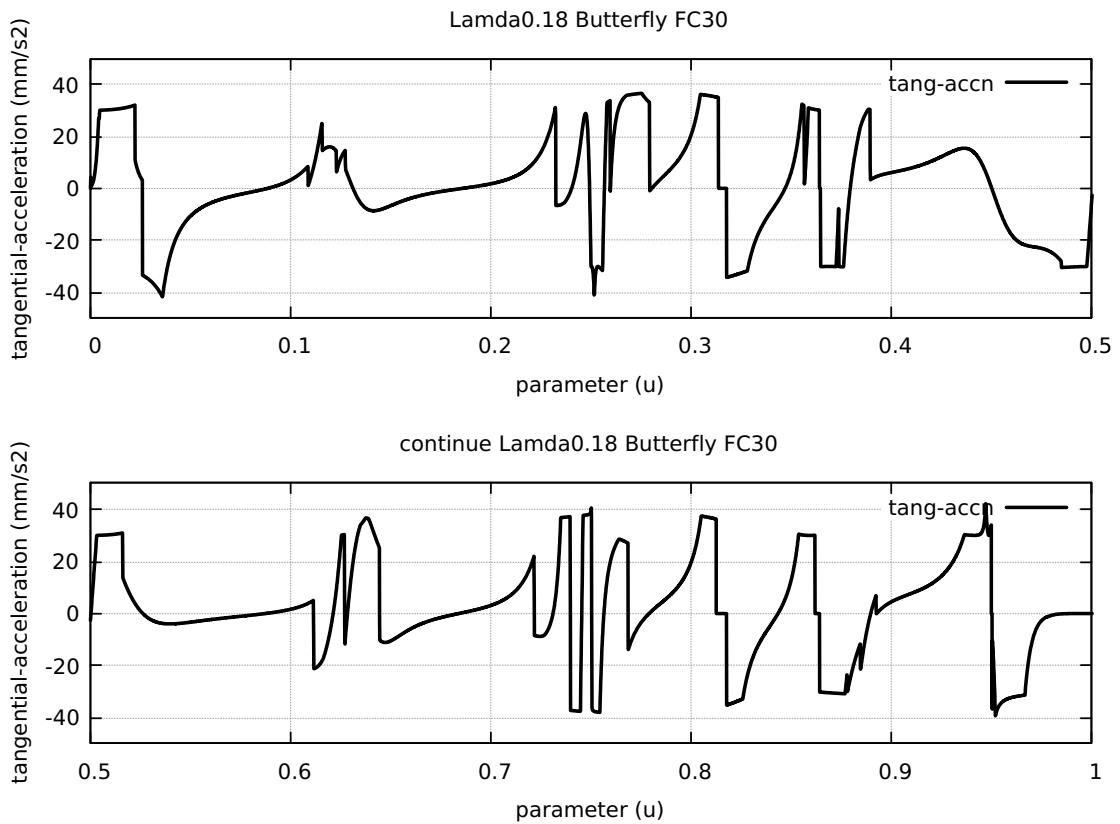


Figure 143: Butterfly FC40 Tangential Acceleration

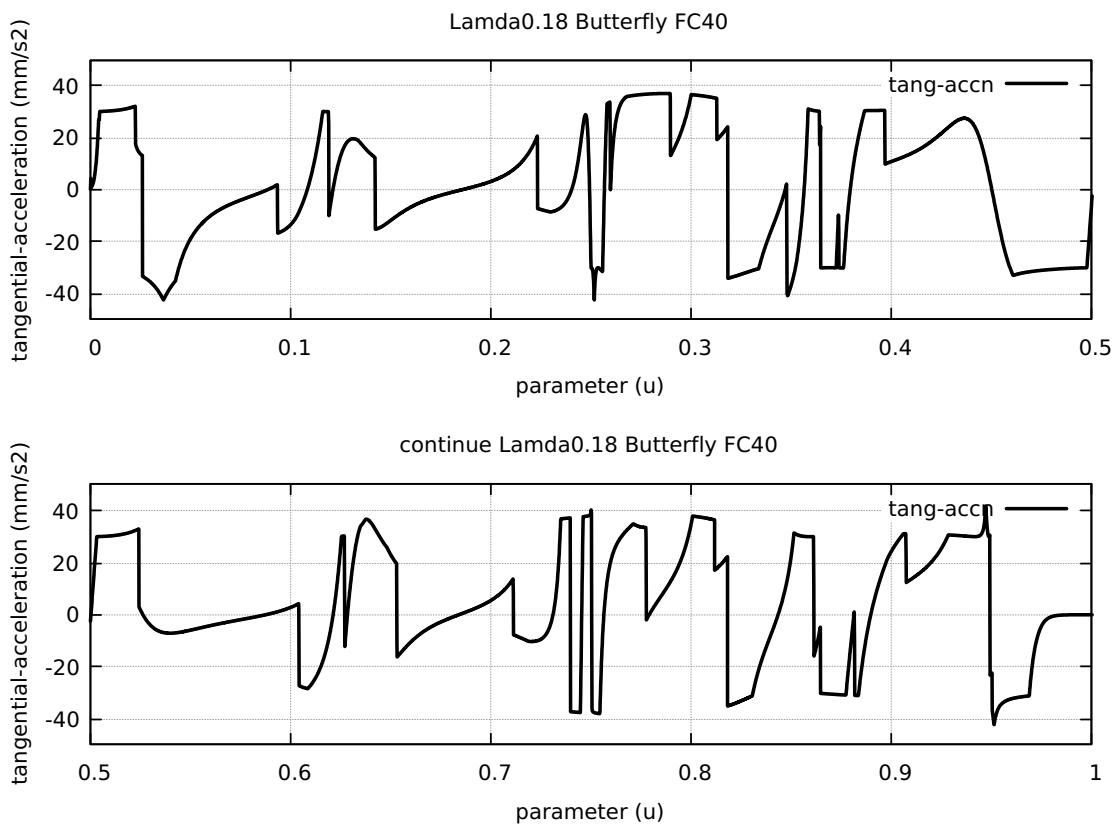


Figure 144: Butterfly FC20 Nominal Separation NAL and NCL

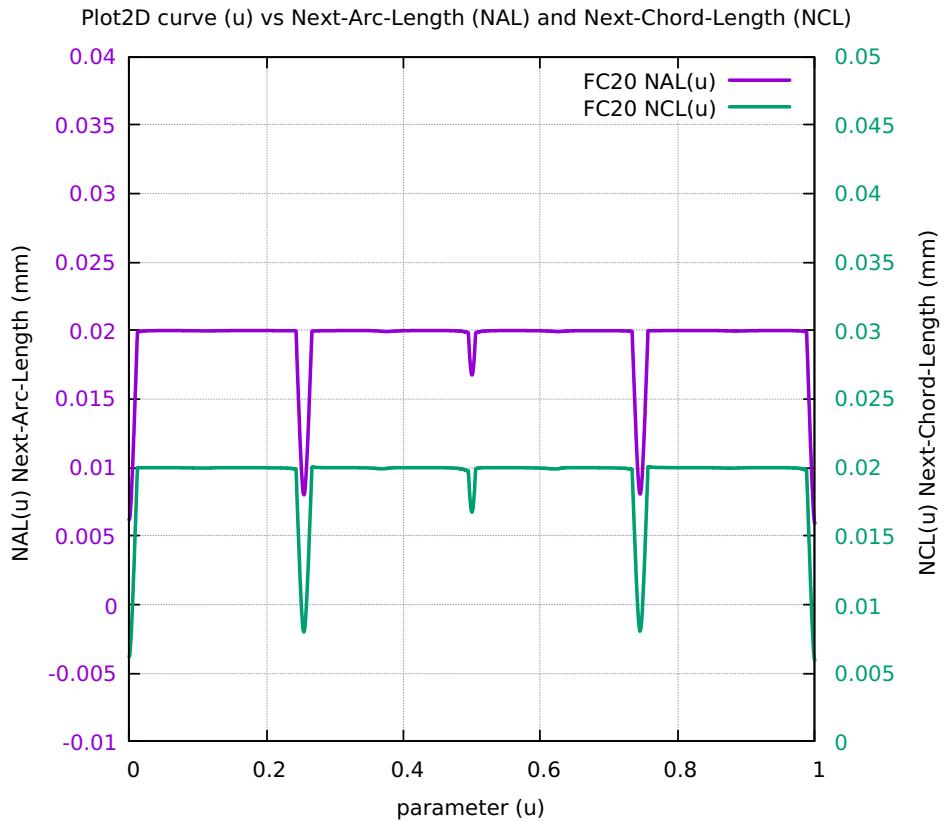


Figure 145: Butterfly Difference SAL minus SCL for FC10 FC20 FC30 FC40

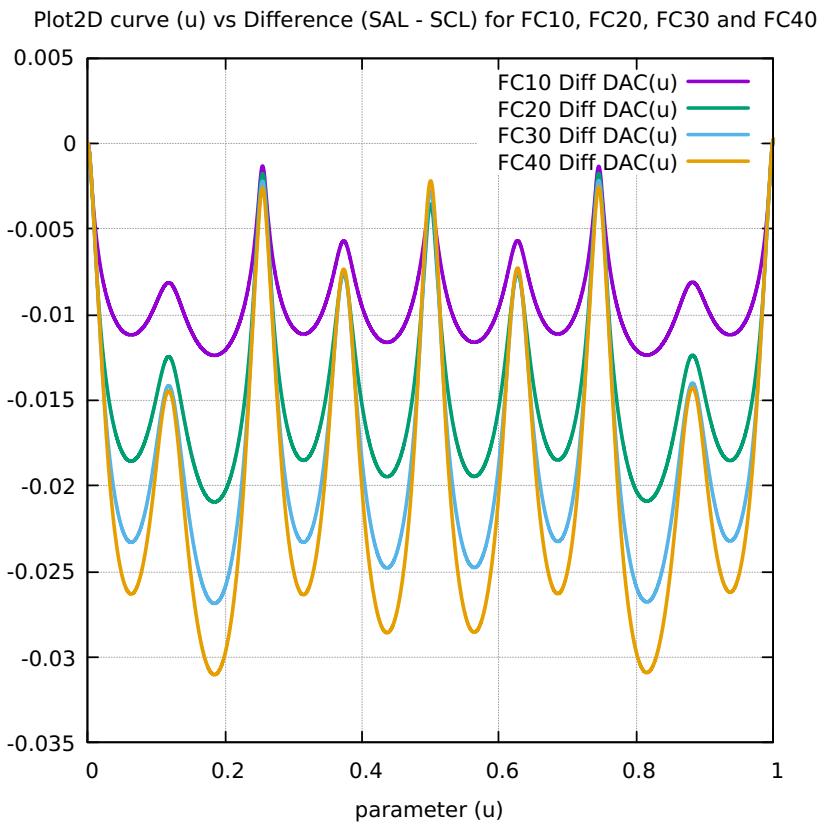


Figure 146: Butterfly FC10 FcateCmd CurrFrate X-Frate Y-Frate

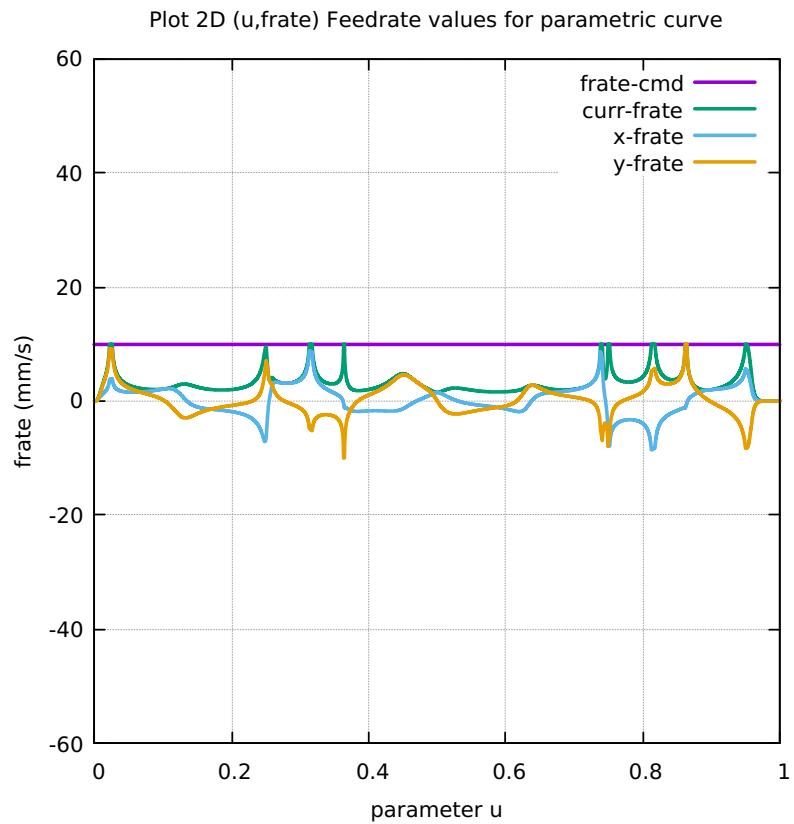


Figure 147: Butterfly FC20 FcateCmd CurrFrate X-Frate Y-Frate

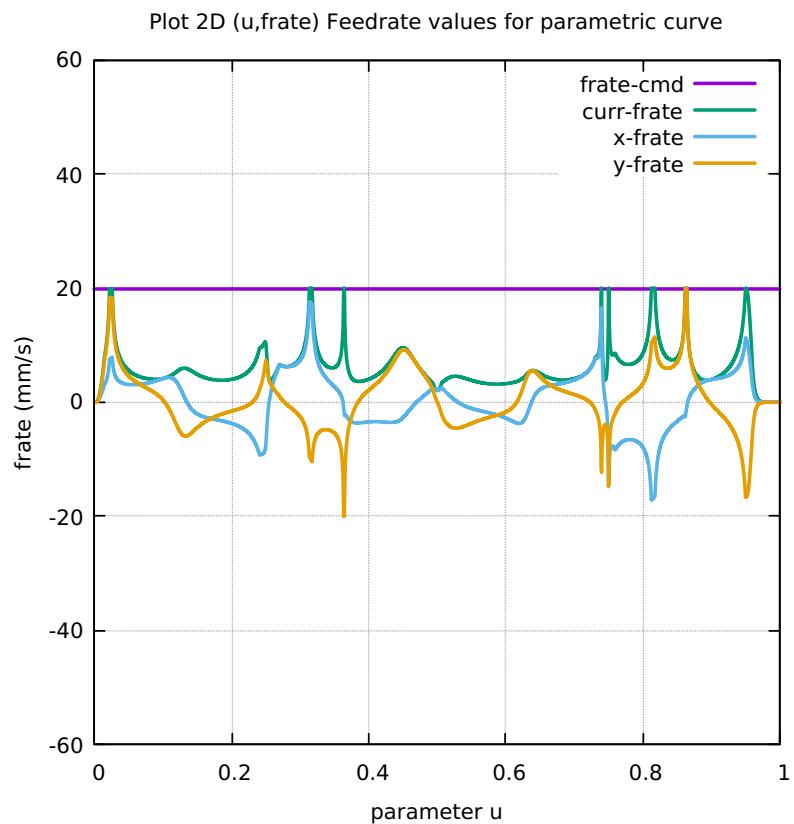


Figure 148: Butterfly FC30 FcateCmd CurrFrate X-Frate Y-Frate

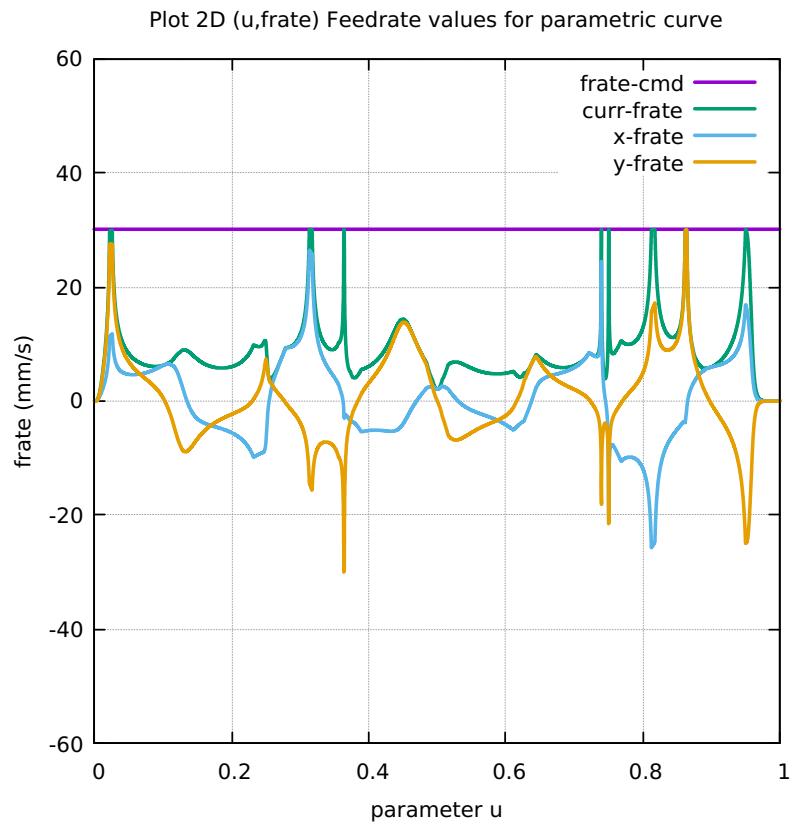


Figure 149: Butterfly FC40 FcateCmd CurrFrate X-Frate Y-Frate

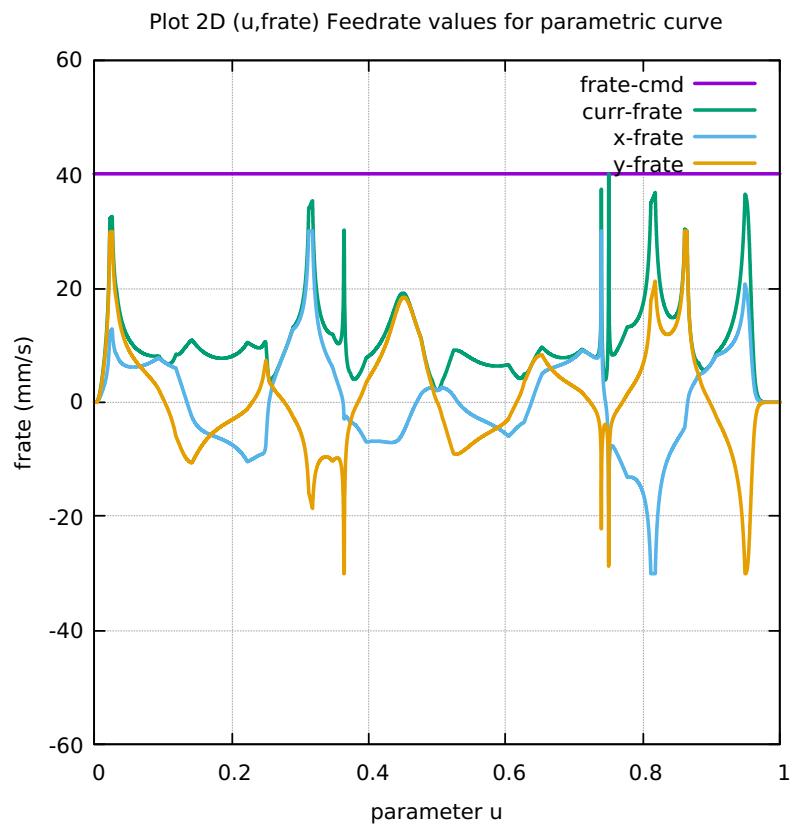


Figure 150: Butterfly FC10 Four Components FeedrateLimit

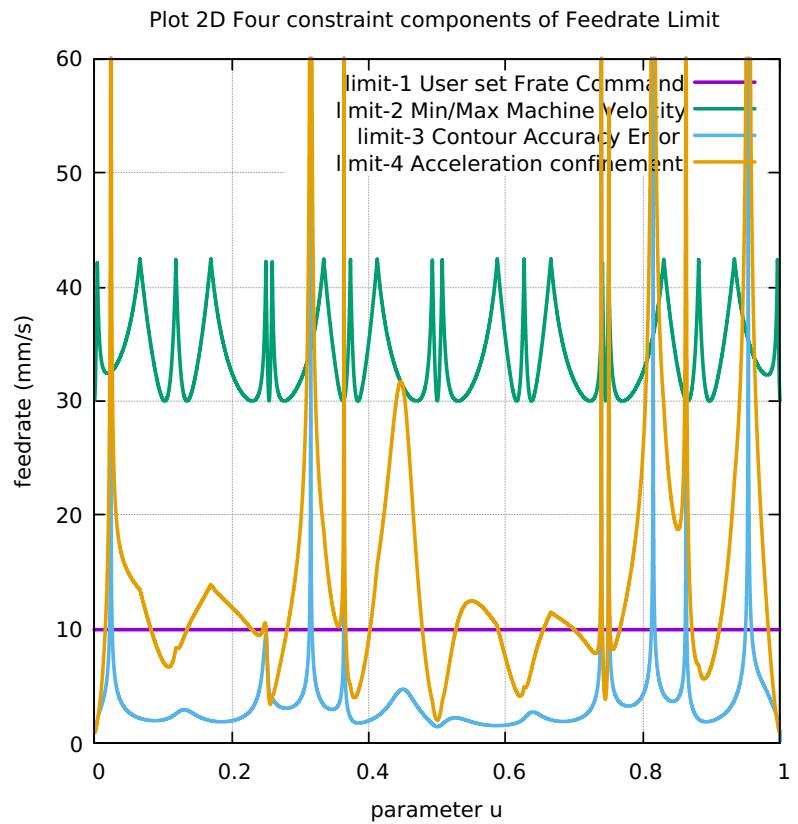


Figure 151: Butterfly FC20 Four Components FeedrateLimit

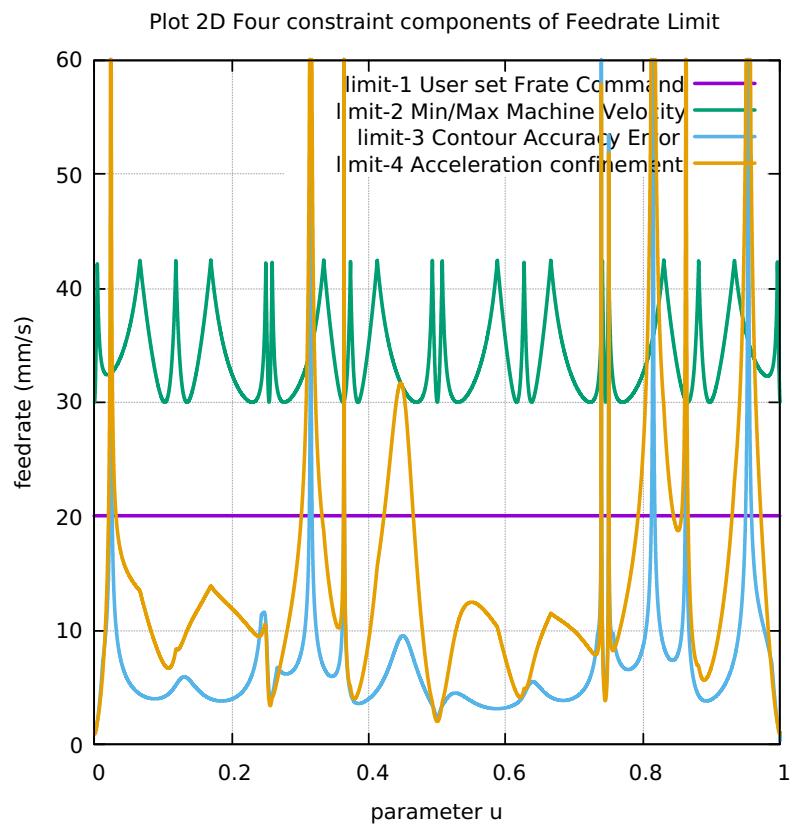


Figure 152: Butterfly FC30 Four Components FeedrateLimit

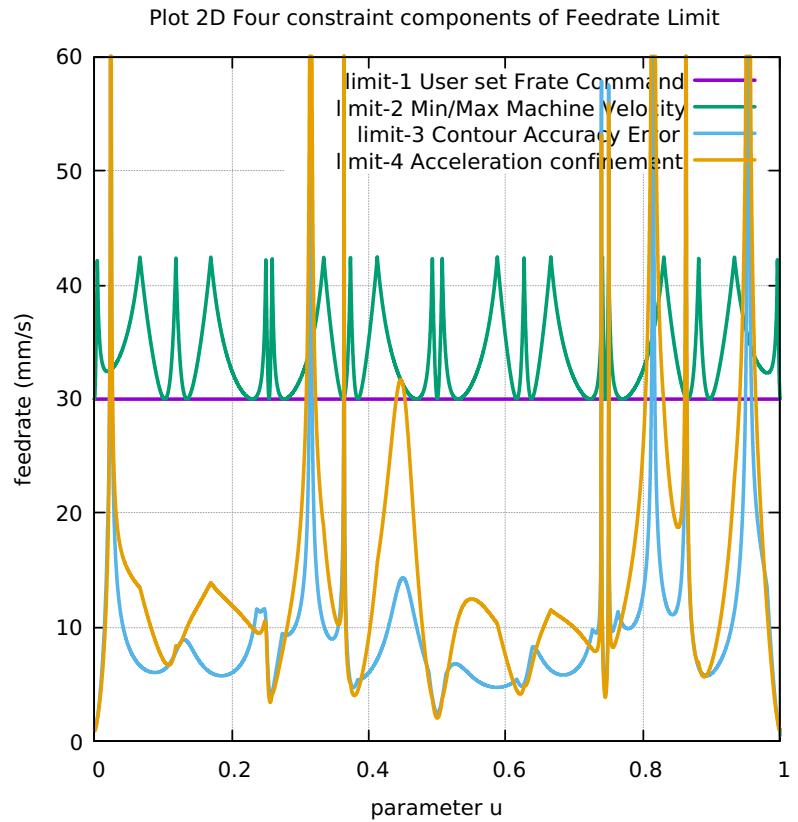


Figure 153: Butterfly FC40 Four Components FeedrateLimit

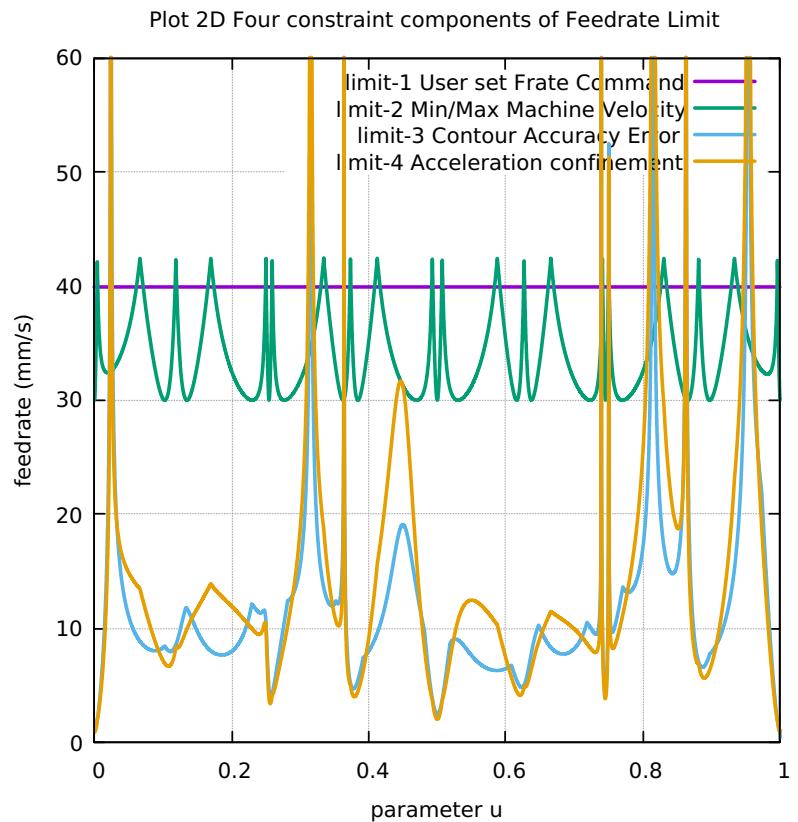


Figure 154: Butterfly Histogram Points FC10 FC20 FC30 FC40

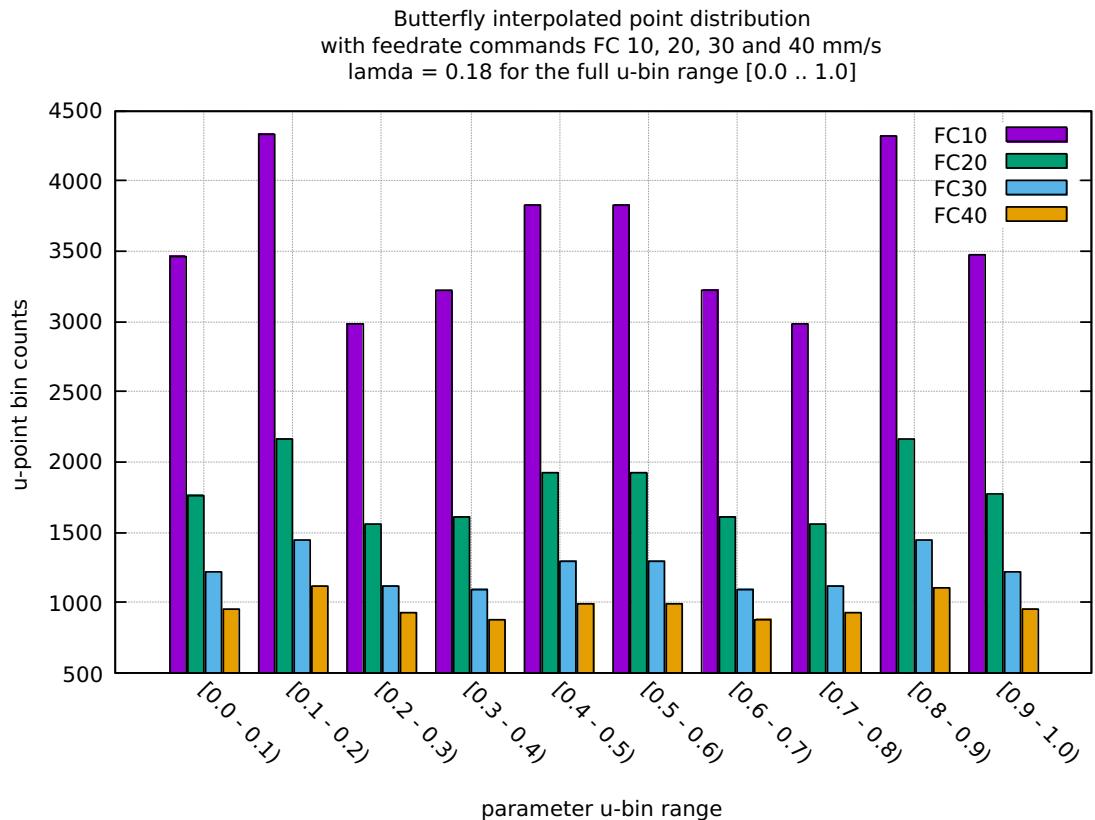


Table 8: Butterfly Table distribution of interpolated points

BINS	FC10	FC20	FC30	FC40
0.0 - 0.1	3463	1763	1214	952
0.1 - 0.2	4332	2167	1444	1112
0.2 - 0.3	2983	1554	1117	927
0.3 - 0.4	3220	1611	1098	877
0.4 - 0.5	3832	1920	1299	998
0.5 - 0.6	3829	1919	1298	997
0.6 - 0.7	3222	1612	1098	878
0.7 - 0.8	2981	1553	1117	926
0.8 - 0.9	4323	2162	1441	1110
0.9 - 1.0	3471	1768	1217	955
Tot Counts	35656	18029	12343	9732

Table 9: Butterfly Table FC10-20-30-40 Run Performance data

1	Curve Type	BUTTERFLY	BUTTERFLY	BUTTERFLY
2	User Feedrate Command FC(mm/s)	FC10 0.18	FC20 0.18	FC40 0.18
3	User Lamda Acceleration Safety Factor			
4	Total Interpolated Points (TIP)	35656	18029	12343
5	Total Sum-Chord-Error (SCE) (mm)	1.938859834664E-03	3.534046223178E-03	4.846582536157E-03
6	Ratio 1 = (SCE/TIP) = Chord-Error/Point	5.437834342068E-08	1.960309642322E-07	3.926902071104E-07
7	Total Sum-Arc-Length (SAL) (mm)	3.560748506870E+02	3.560738974072E+02	3.560730284349E+02
8	Total Sum-Chord-Length (SCL) (mm)	3.560747025702E+02	3.560737108999E+02	3.560727930088E+02
9	Difference = (SAL - SCL) (mm)	1.481168304736E-04	1.865072539431E-04	2.354260789161E-04
10	Percentage Difference = (SAL - SCL)/SAL	4.159710526812E-05	5.237880543932E-05	6.611735799000E-05
11	Ratio 2 = (SCE/SCL) = Chord Error/Chord-Length	5.445092899523E-06	9.925041122094E-06	1.361121273884E-05
12	Total Sum-Arc-Theta (SAT) (rad)	2.212404302532E+01	2.212055105865E+01	2.211618559661E+01
13	Total Sum-Arc-Area (SAA) (mm2)	1.758667624644E-04	6.462621773264E-04	1.298932073590E-03
14	Ratio 3 = (SAA/SCL) = Arc-Area/Chord-Length	5.445092899523E-06	9.925041122094E-06	1.361121273884E-05
15	Average-Chord-Error (ACE) (mm)	5.437834342068E-08	1.960309642322E-07	3.926902071104E-07
16	Average-Arc-Length (AAL) (mm)	9.986673697574E-03	1.975115916392E-02	2.885051275603E-02
17	Average-Chord-Length (ACL) (mm)	9.986669543407E-03	1.975114881850E-02	2.885049368083E-02
18	Average-Arc-Theta (AAT) (rad)	6.205032400874E-04	1.227010819761E-03	1.791945032946E-03
19	Average-Arc-Area (AAA) (mm2)	4.932457228003E-09	3.584769122068E-08	1.052448609294E-07
20	Algorithm actual runtime on computer (ART) (s)	14.219218872	8.1882168602	8.667691811
				10.542664043

.7 APPENDIX SNAILSHELL CURVE

- .7.1 Plot of Snailshell curve [155]**
- .7.2 Snailshell Radius of Curvature [156]**
- .7.3 Snailshell Validation in LinuxCNC [157]**
- .7.4 Snailshell Direction of Travel 3D [158]**
- .7.5 Snailshell First and Second Order Taylor's Approx [159]**
- .7.6 Snailshell First minus Second Order Taylor's Approx [160]**
- .7.7 Snailshell Separate First Second Order Taylor's Approx [161]**
- .7.8 Snailshell Separation SAL and SCL [162]**
- .7.9 Snailshell Chord-error in close view 2 scales [163]**
- .7.10 Snailshell Four Components Feedrate Limit [164]**
- .7.11 Snailshell FrateCommand FrateLimit and Curr-Frate [165]**
- .7.12 Snailshell FeedRateLimit minus CurrFeedRate [166]**
- .7.13 Snailshell FC20-Nominal X and Y Feedrate Profiles [167]**
- .7.14 Snailshell FC20 Nominal Tangential Acceleration [168]**
- .7.15 Snailshell FC20 Nominal Rising S-Curve Profile [169]**
- .7.16 Snailshell FC20 Nominal Falling S-Curve Profile [170]**
- .7.17 Snailshell FC10 Colored Feedrate Profile data ngcode [171]**
- .7.18 Snailshell FC20 Colored Feedrate Profile data ngcode [172]**

- .7.19 Snailshell FC30 Colored Feedrate Profile data ngcode [173]
- .7.20 Snailshell FC40 Colored Feedrate Profile data ngcode [174]
- .7.21 Snailshell FC10 Tangential Acceleration [175]
- .7.22 Snailshell FC20 Tangential Acceleration [176]
- .7.23 Snailshell FC30 Tangential Acceleration [177]
- .7.24 Snailshell FC40 Tangential Acceleration [178]
- .7.25 Snailshell FC20 Nominal Separation NAL and NCL [179]
- .7.26 Snailshell SAL minus SCL for FC10 FC20 FC30 FC40 [180]
- .7.27 Snailshell FC10 FrateCmd CurrFrate X-Frate Y-Frate [181]
- .7.28 Snailshell FC20 FrateCmd CurrFrate X-Frate Y-Frate [182]
- .7.29 Snailshell FC30 FrateCmd CurrFrate X-Frate Y-Frate [183]
- .7.30 Snailshell FC40 FrateCmd CurrFrate X-Frate Y-Frate [184]
- .7.31 Snailshell FC10 Four Components FeedrateLimit [185]
- .7.32 Snailshell FC20 Four Components FeedrateLimit [186]
- .7.33 Snailshell FC30 Four Components FeedrateLimit [187]
- .7.34 Snailshell FC40 Four Components FeedrateLimit [188]
- .7.35 Snailshell Histogram Points FC10 FC20 FC30 FC40 [189]
- .7.36 Snailshell Table distribution of interpolated points [10]
- .7.37 Snailshell Table FC10-20-30-40 Run Performance data [11]

Figure 155: Plot of Snailshell curve

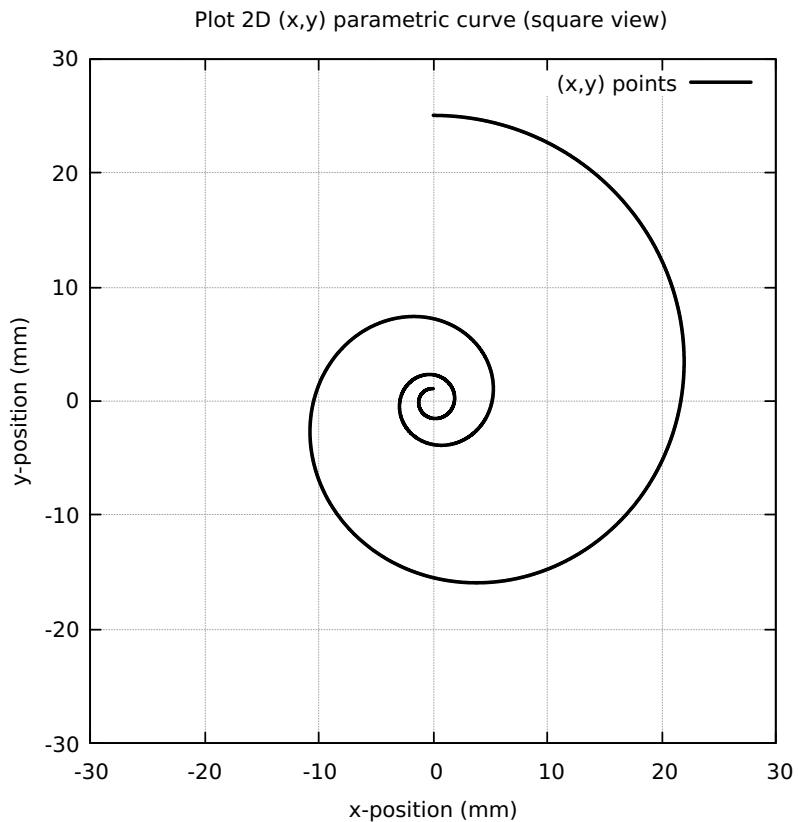


Figure 156: Snailshell Radius of Curvature

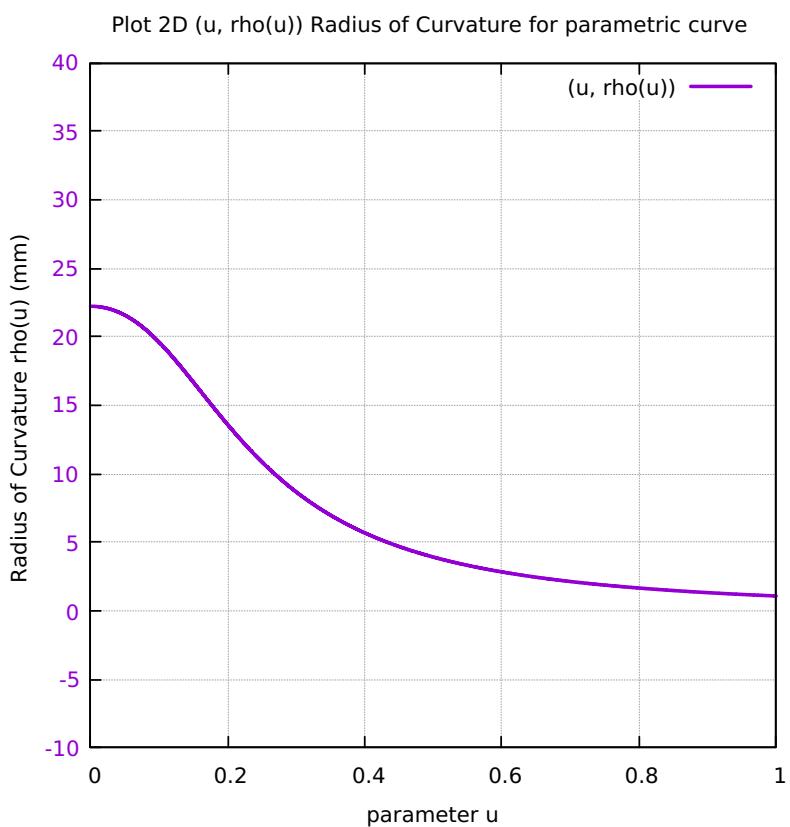


Figure 157: Snailshell Validation in LinuxCNC

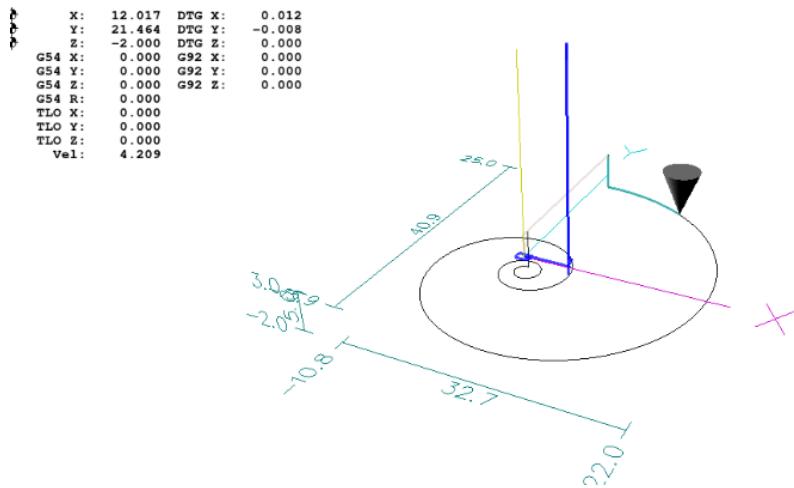


Figure 158: Snailshell Direction of Travel 3D

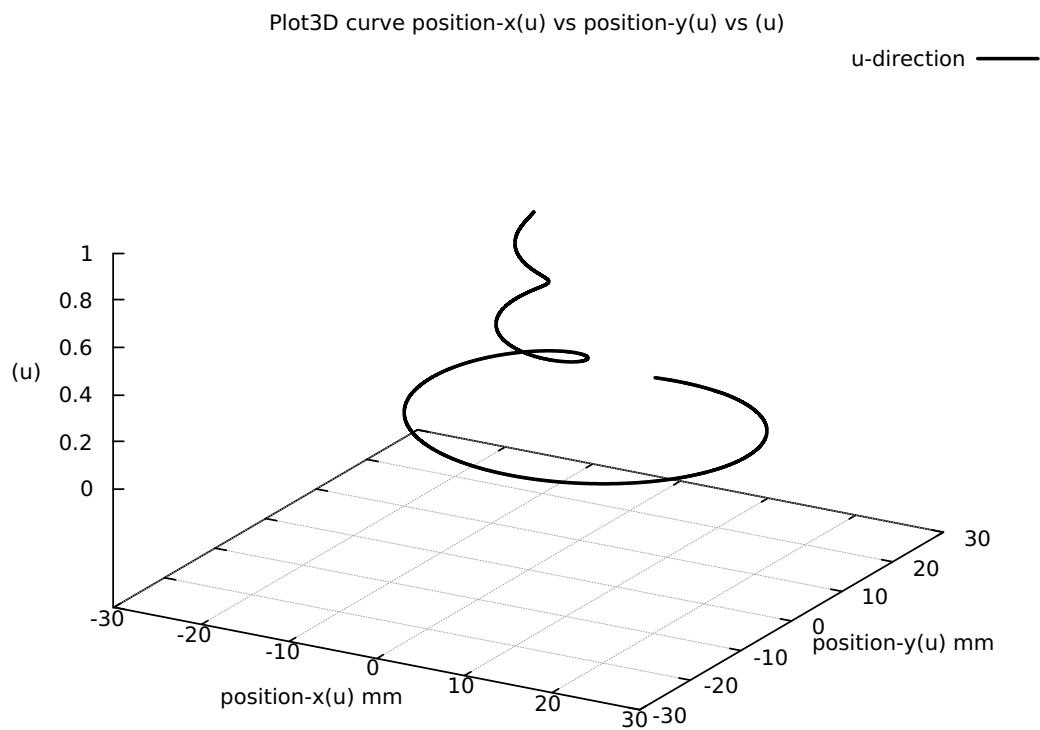


Figure 159: Snailshell First and Second Order Taylor's Approximation

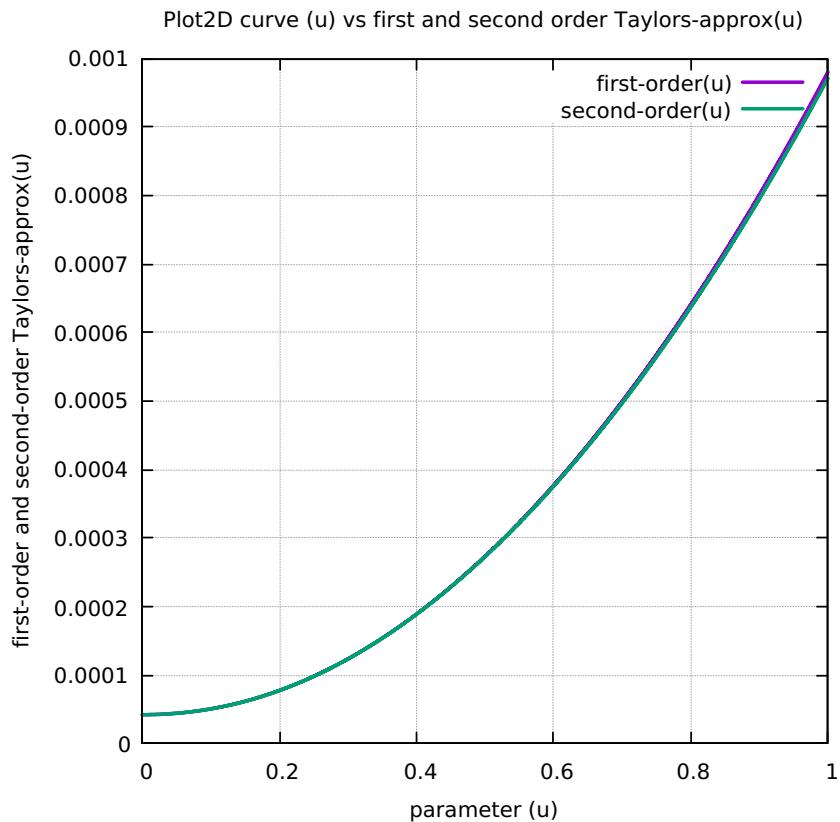


Figure 160: Snailshell First minus Second Order Taylor's Approximation

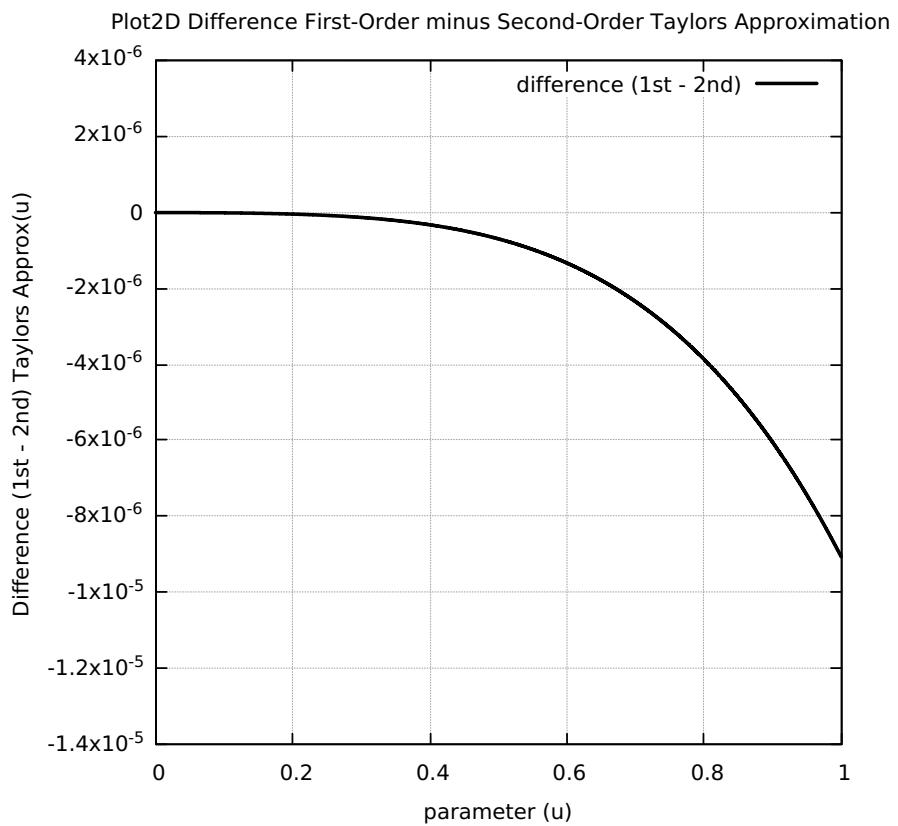


Figure 161: Snailshell Separation First and Second Order Taylor's Approximation

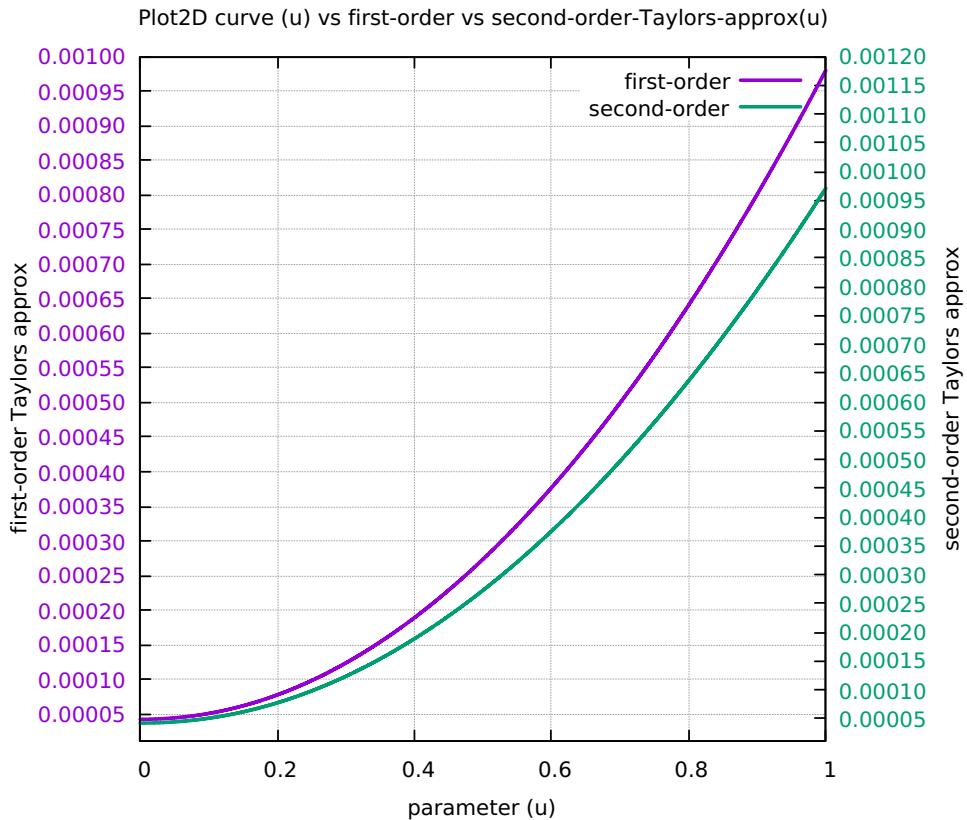


Figure 162: Snailshell Separation SAL and SCL

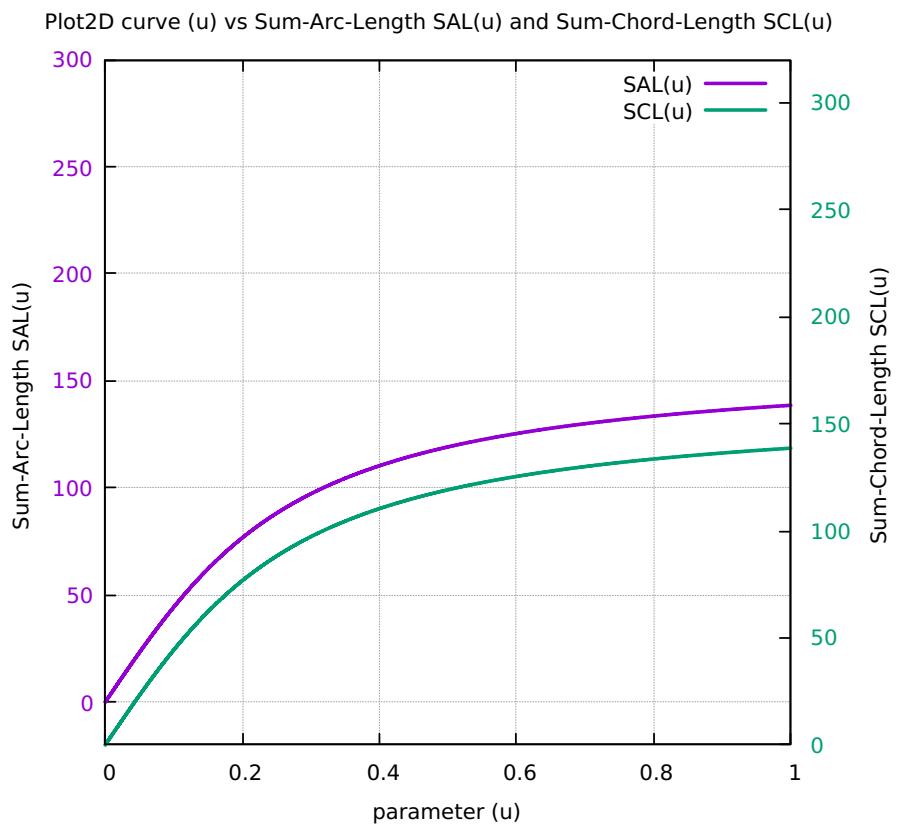


Figure 163: Snailshell Chord-error in close view 2 scales

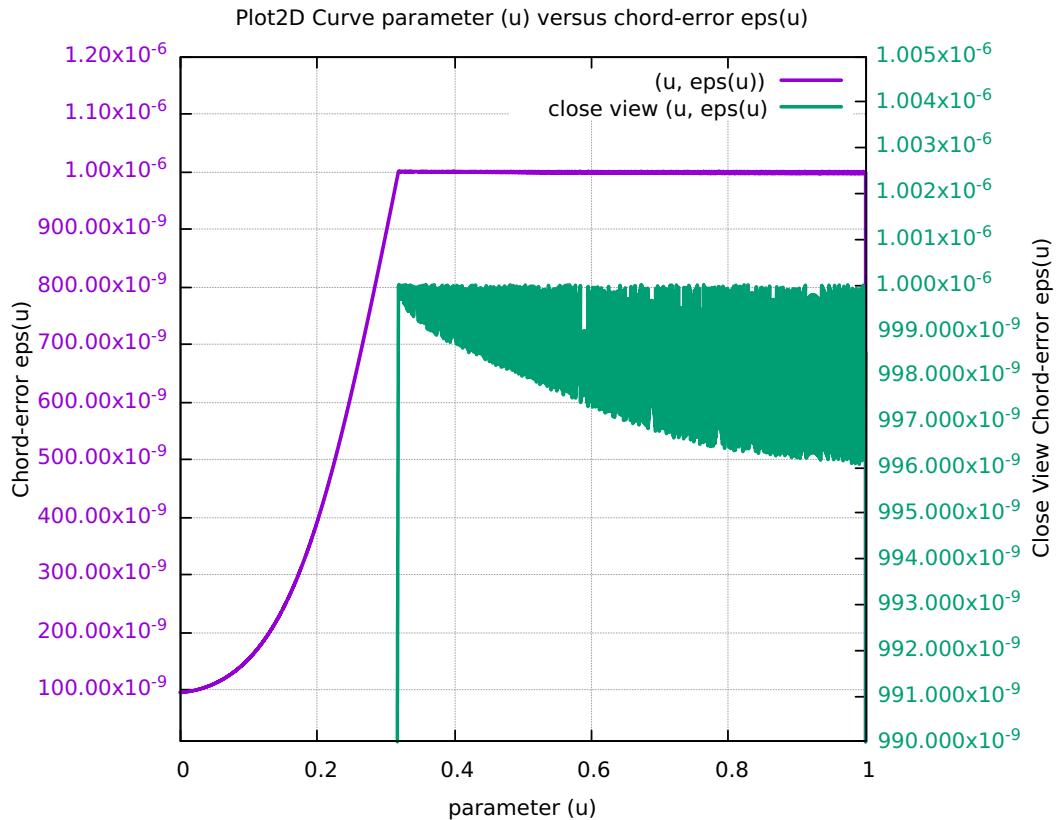


Figure 164: Snailshell Four Components Feedrate Limit

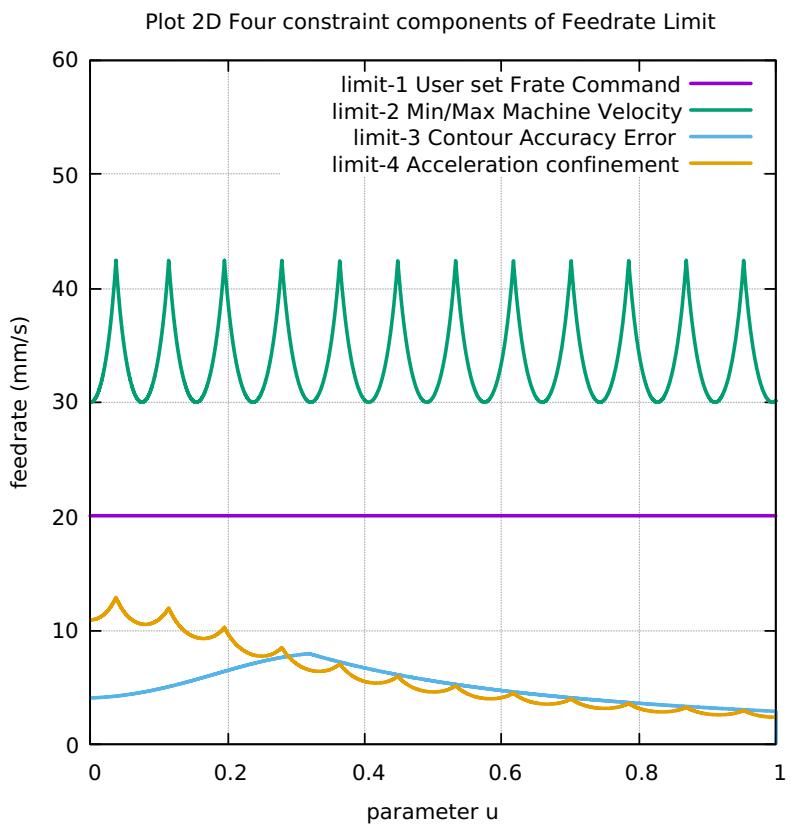


Figure 165: Snailshell FrateCommand FrateLimit and Curr-Frate

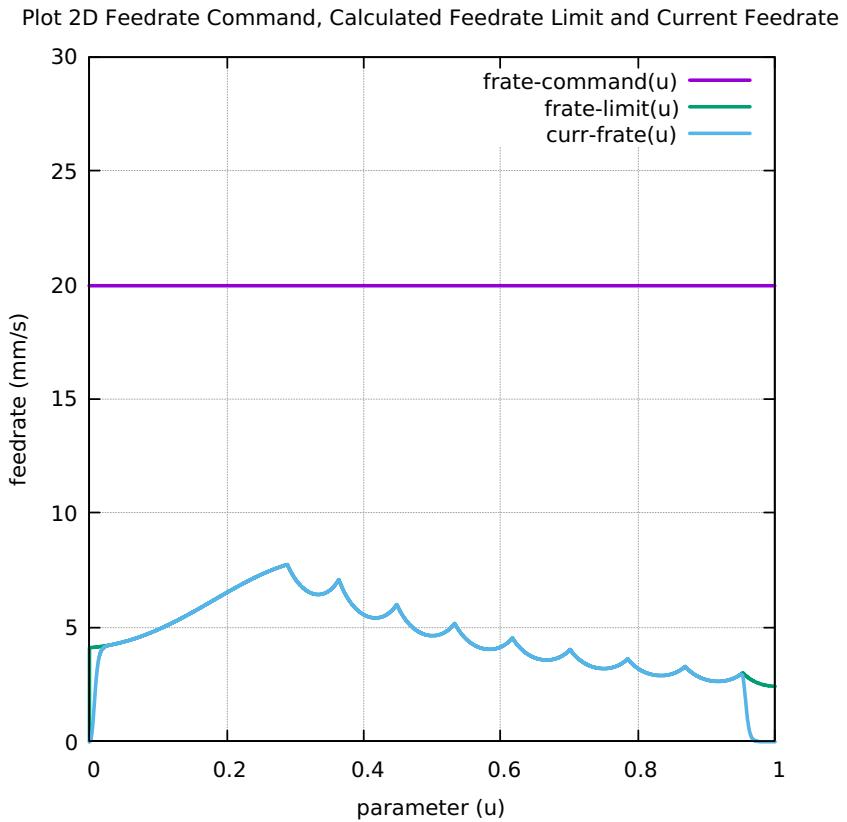


Figure 166: Snailshell FeedRateLimit minus CurrFeedRate

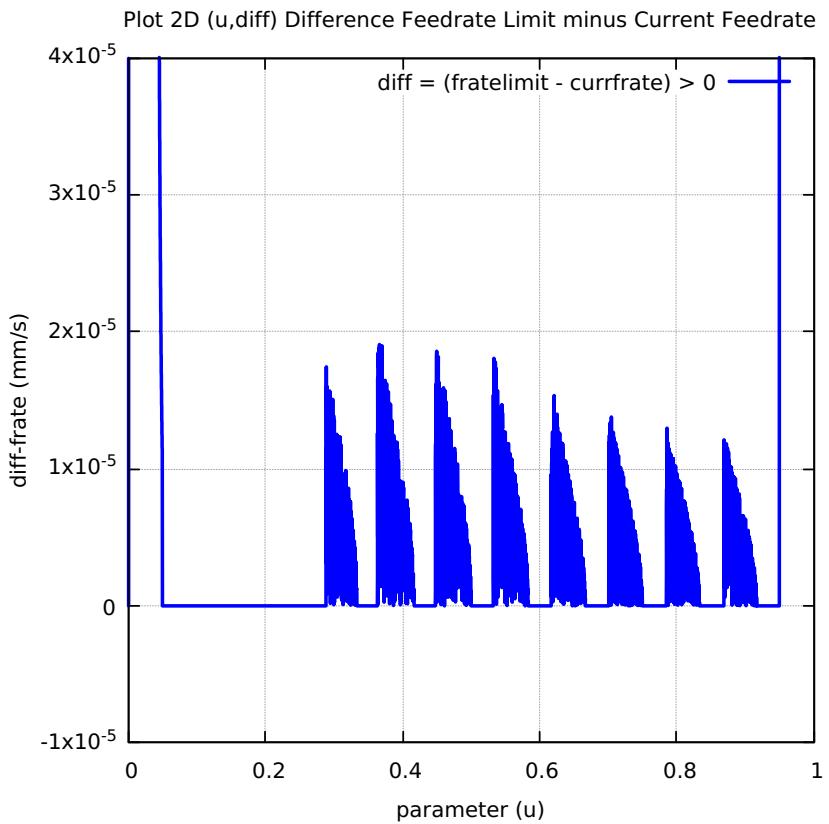


Figure 167: Snailshell FC20-Nominal X and Y Feedrate Profiles

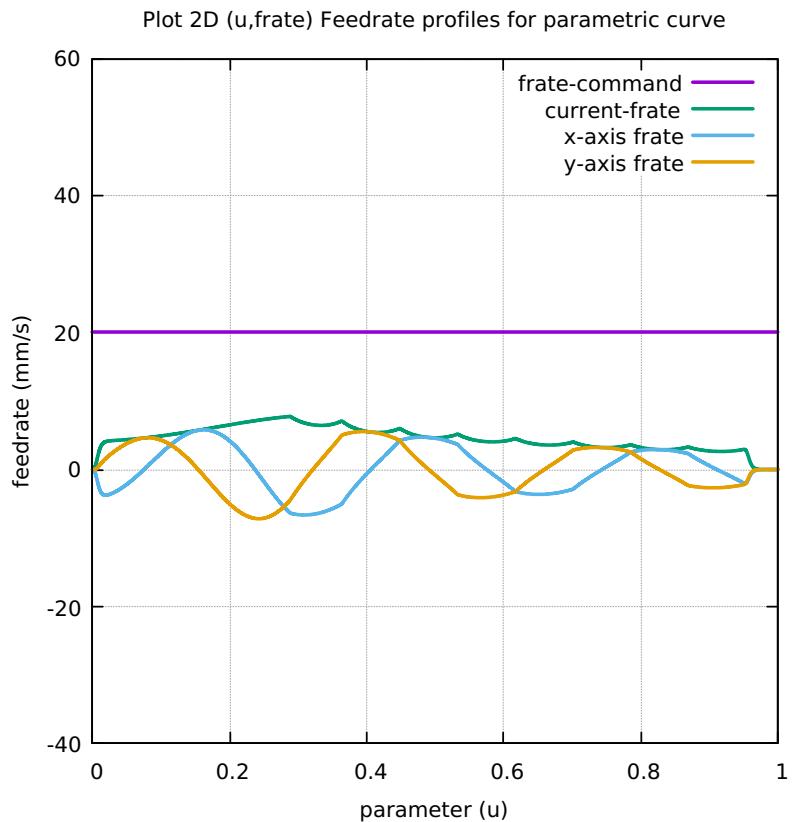


Figure 168: Snailshell FC20 Nominal Tangential Acceleration

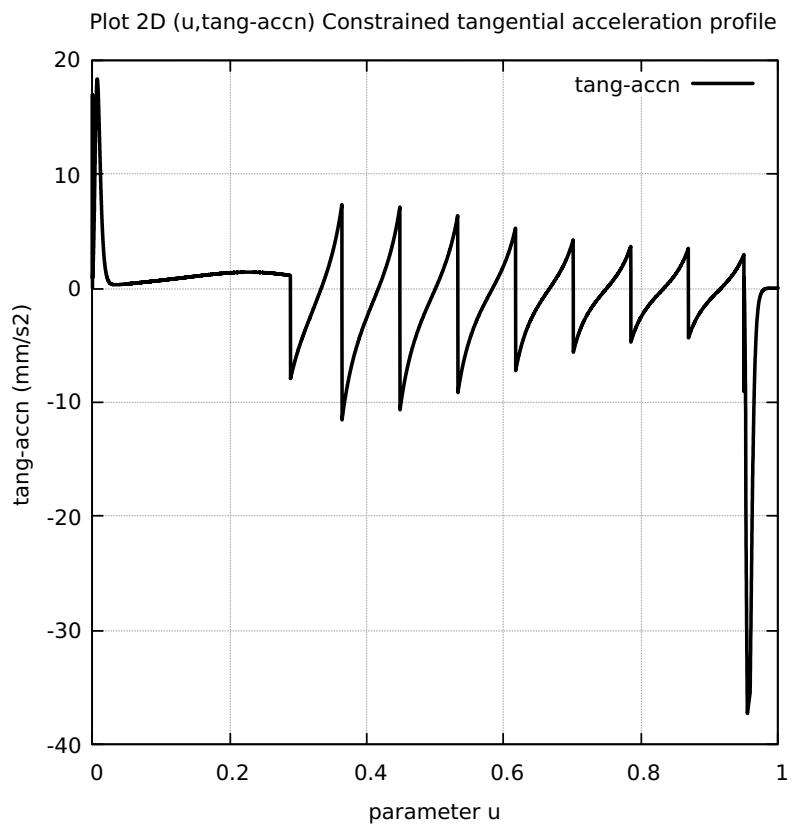


Figure 169: Snailshell FC20 Nominal Rising S-Curve Profile

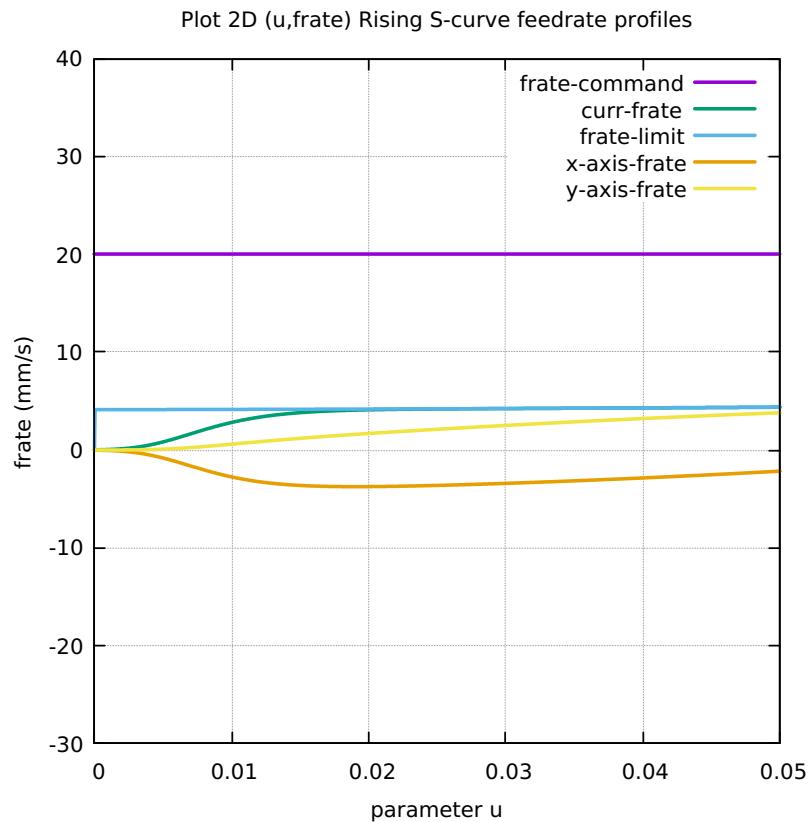


Figure 170: Snailshell FC20 Nominal Falling S-Curve Profile

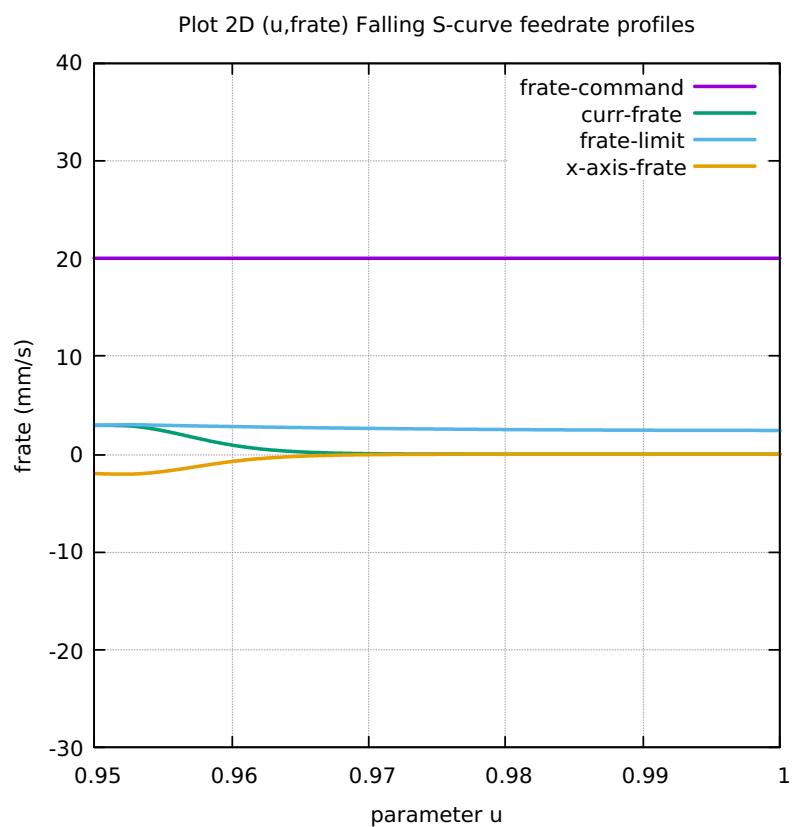


Figure 171: Snailshell FC10 Colored Feedrate Profile data ngcode

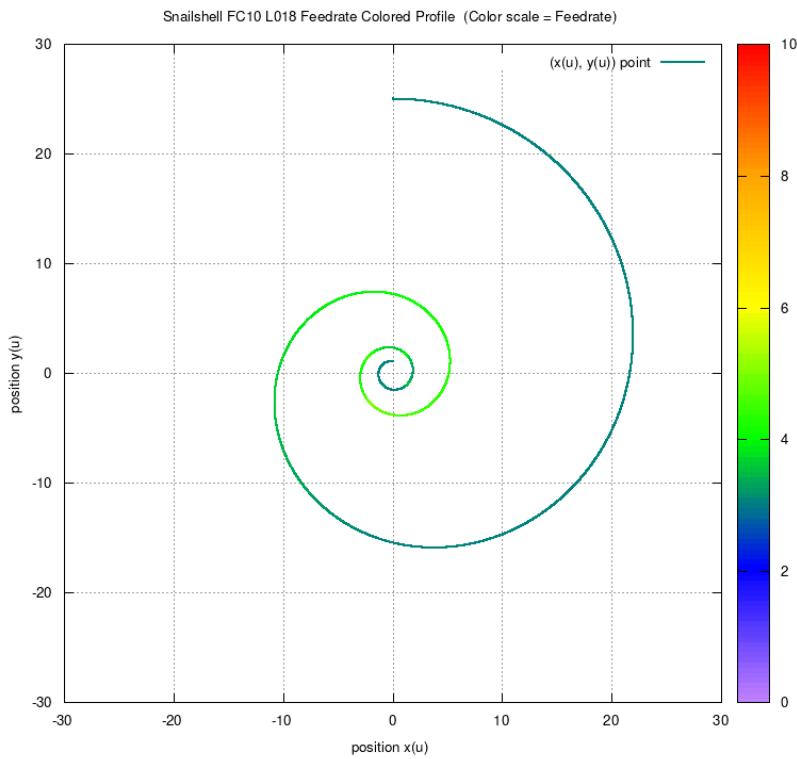


Figure 172: Snailshell FC20 Colored Feedrate Profile data ngcode

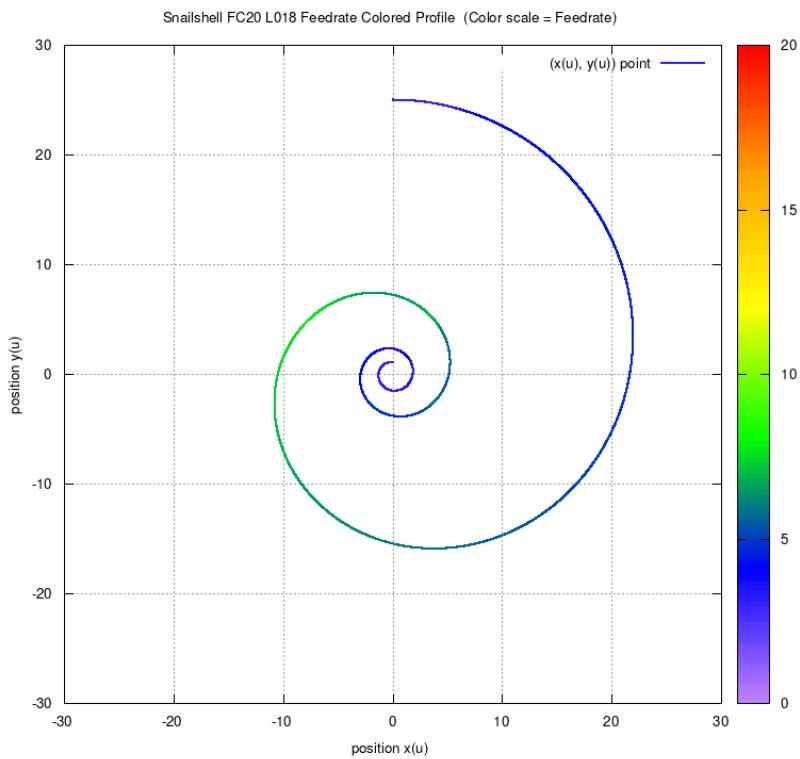


Figure 173: Snailshell FC30 Colored Feedrate Profile data ngcode

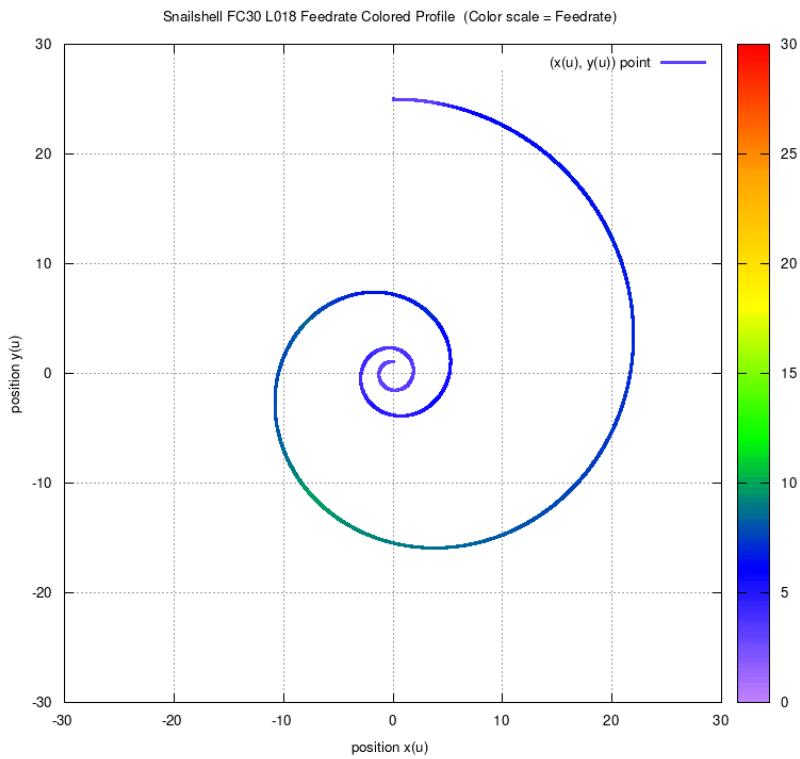


Figure 174: Snailshell FC40 Colored Feedrate Profile data ngcode

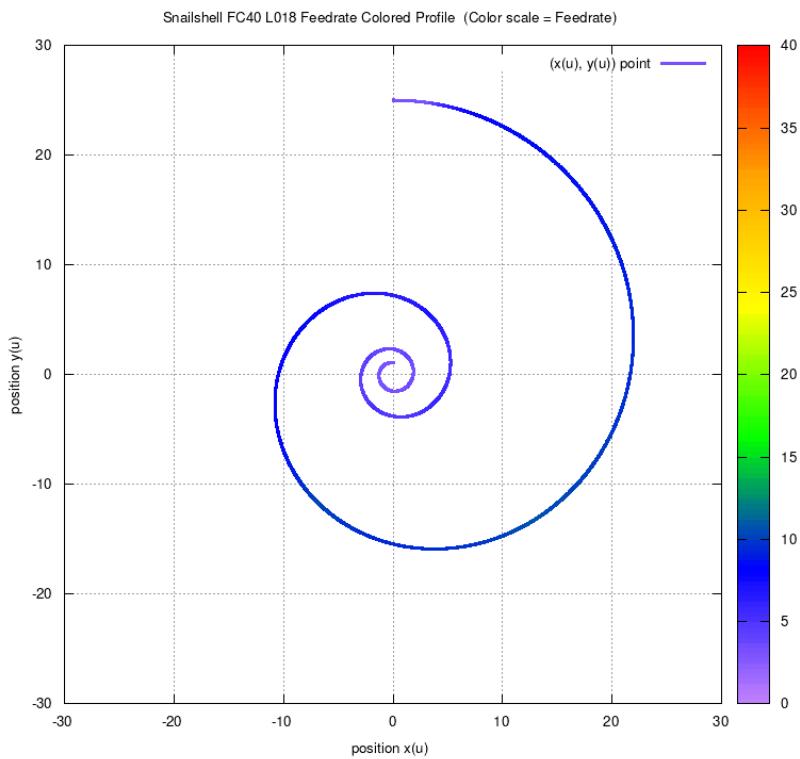


Figure 175: Snailshell FC10 Tangential Acceleration

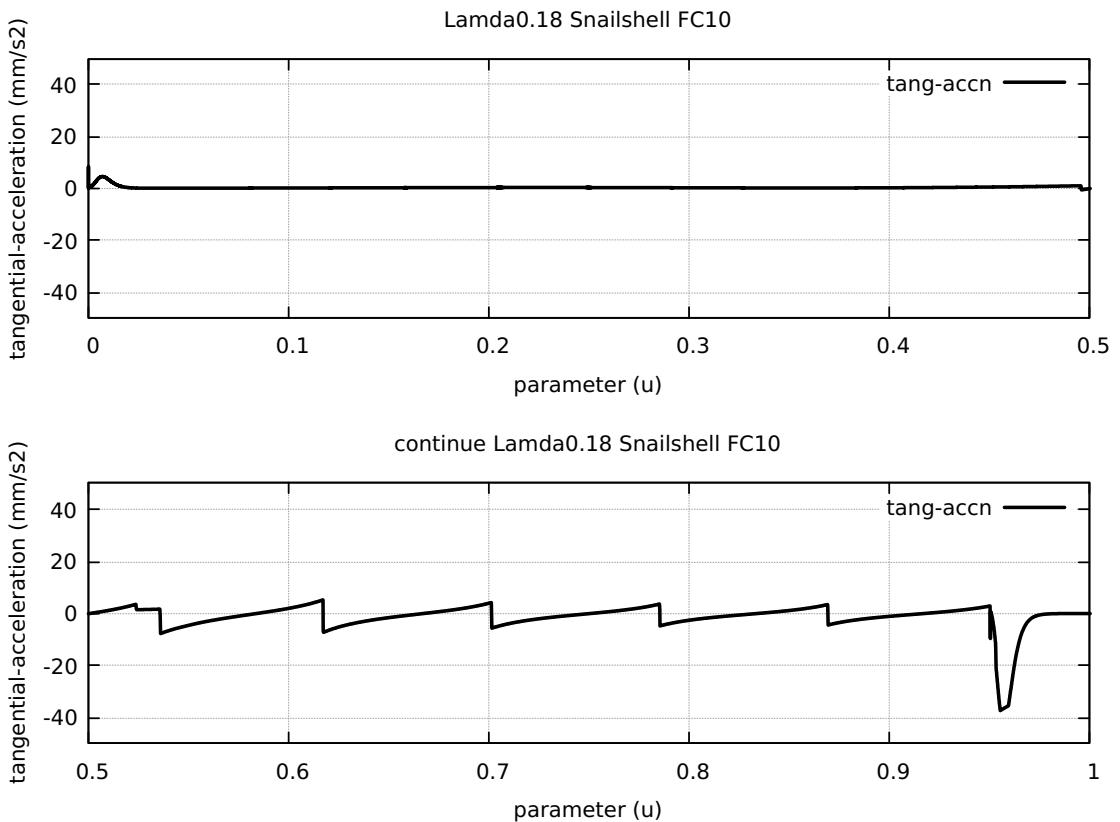


Figure 176: Snailshell FC20 Tangential Acceleration

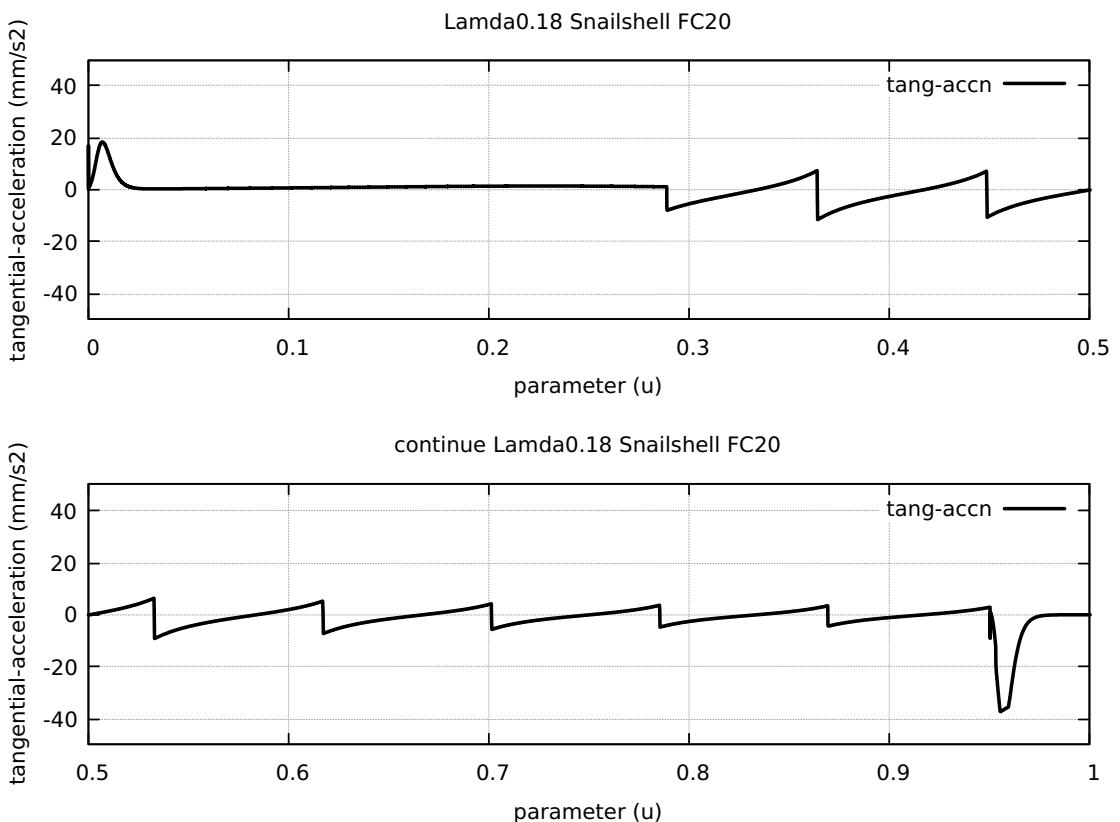


Figure 177: Snailshell FC30 Tangential Acceleration

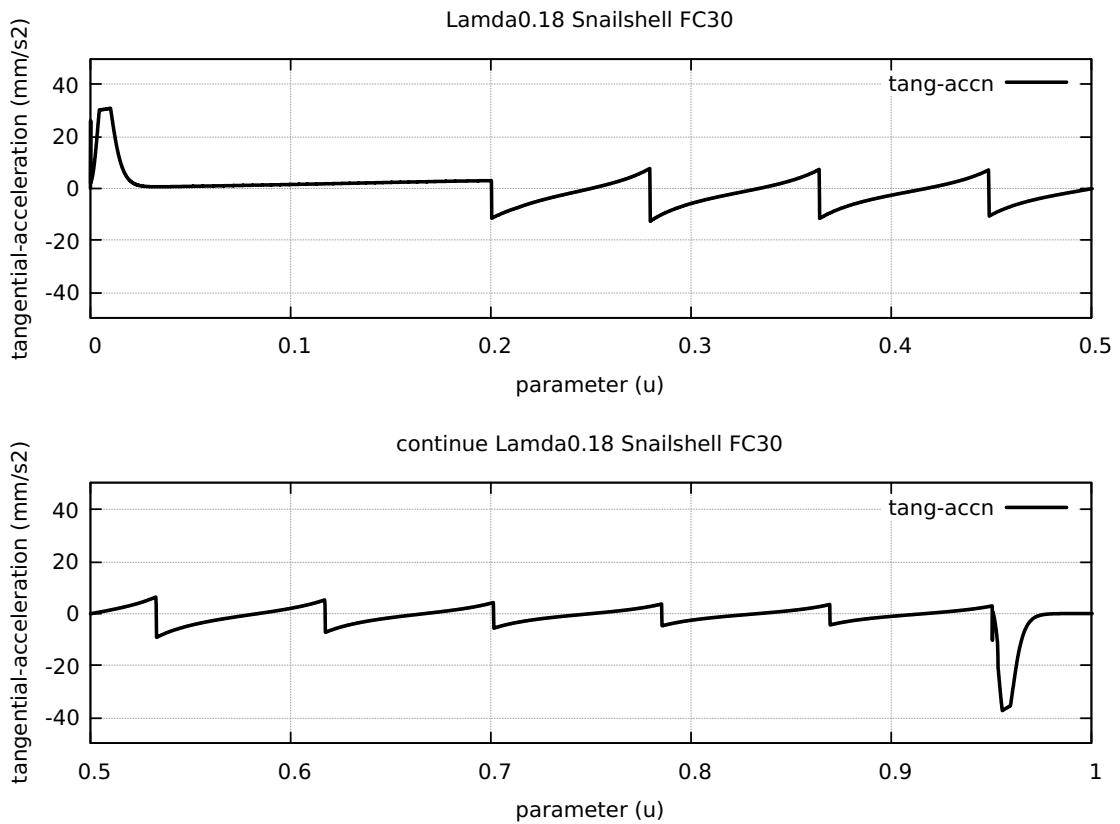


Figure 178: Snailshell FC40 Tangential Acceleration

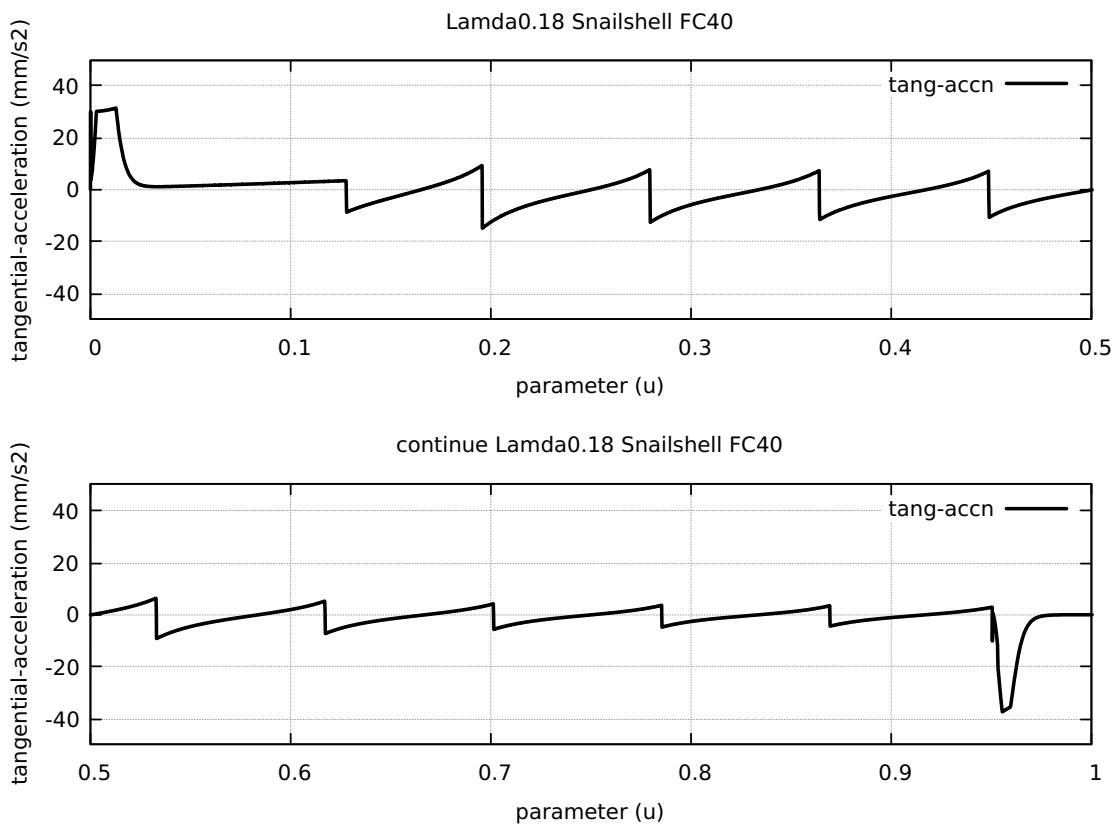


Figure 179: Snailshell FC20 Nominal Separation NAL and NCL

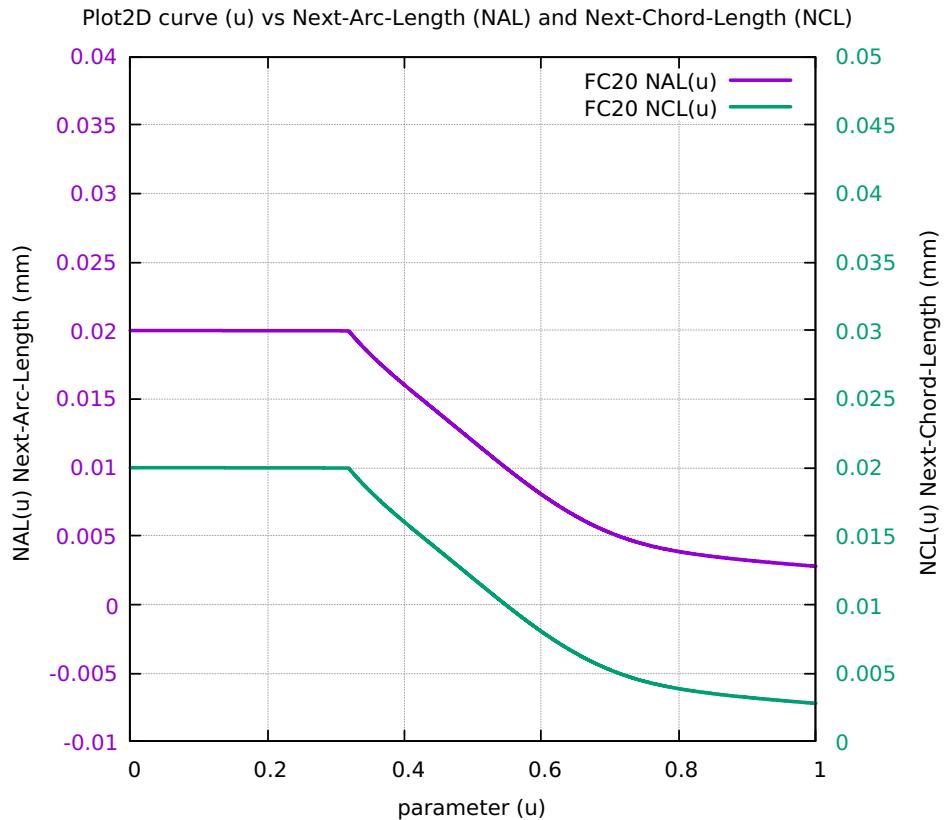


Figure 180: Snailshell Difference SAL minus SCL for FC10 FC20 FC30 FC40

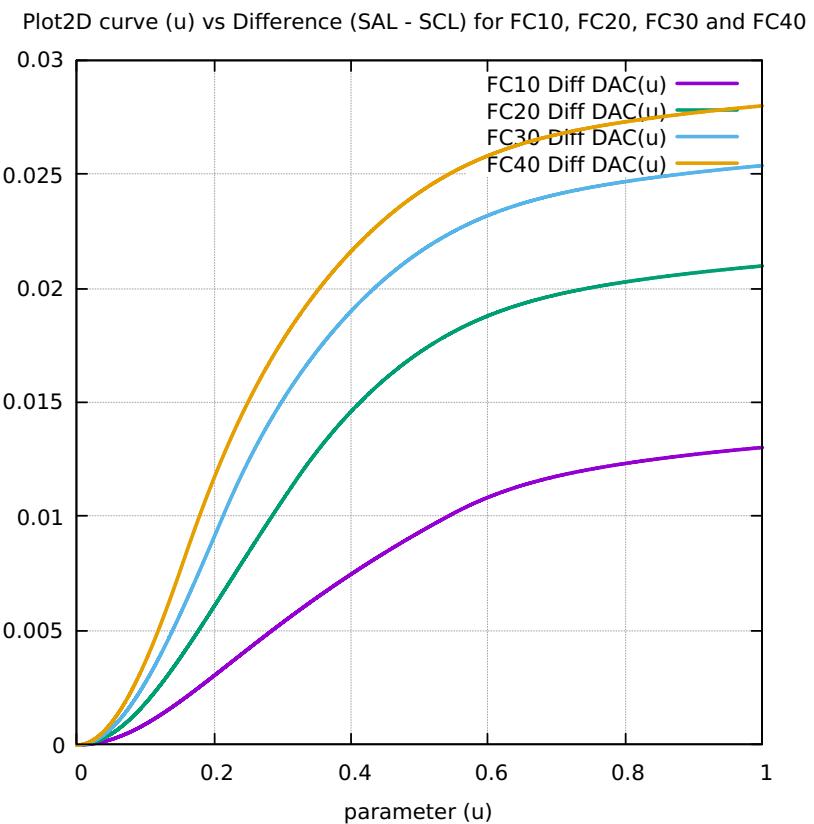


Figure 181: Snailshell FC10 FrateCmd CurrFrate X-Frate Y-Frate

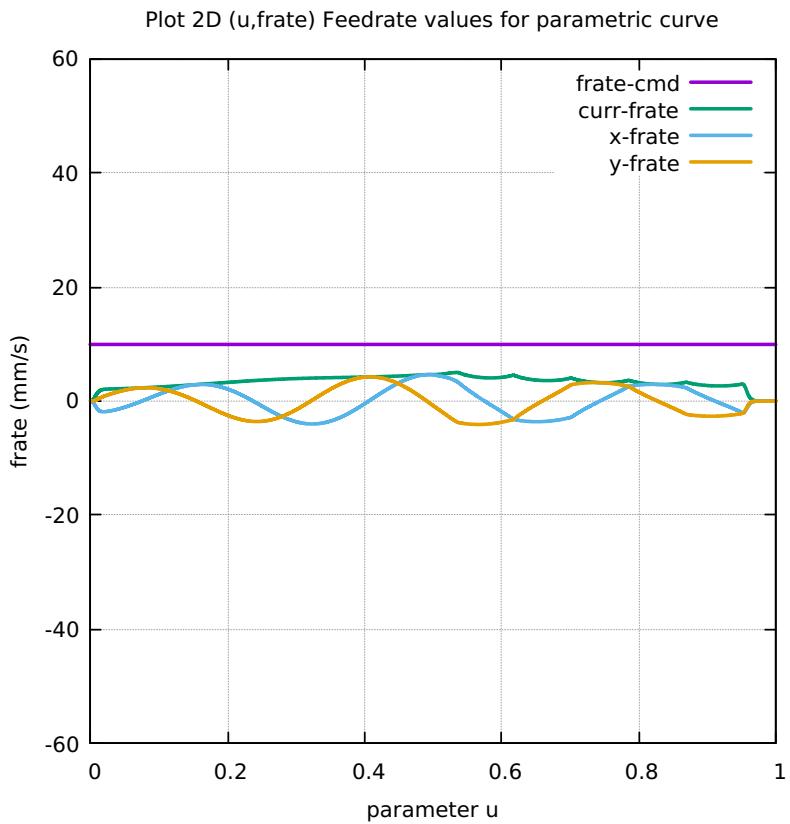


Figure 182: Snailshell FC20 FrateCmd CurrFrate X-Frate Y-Frate

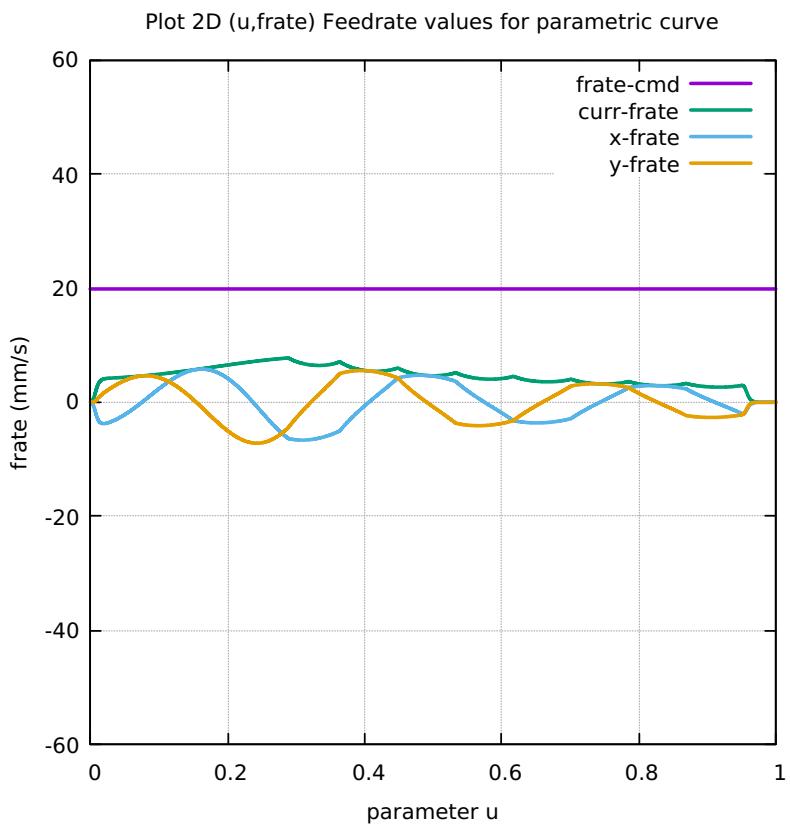


Figure 183: Snailshell FC30 FrateCmd CurrFrate X-Frate Y-Frate

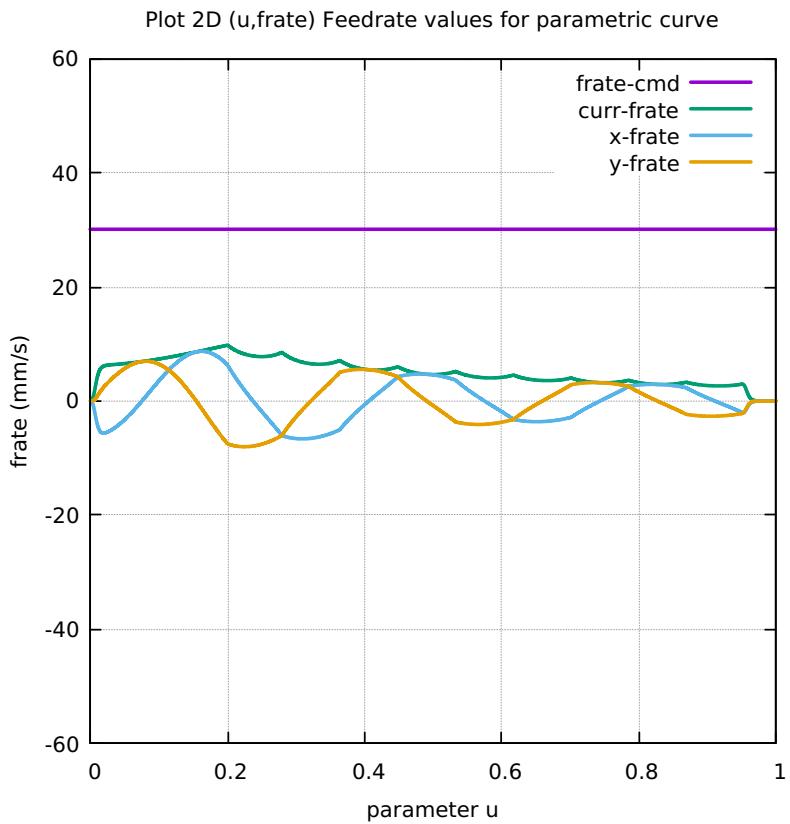


Figure 184: Snailshell FC40 FrateCmd CurrFrate X-Frate Y-Frate

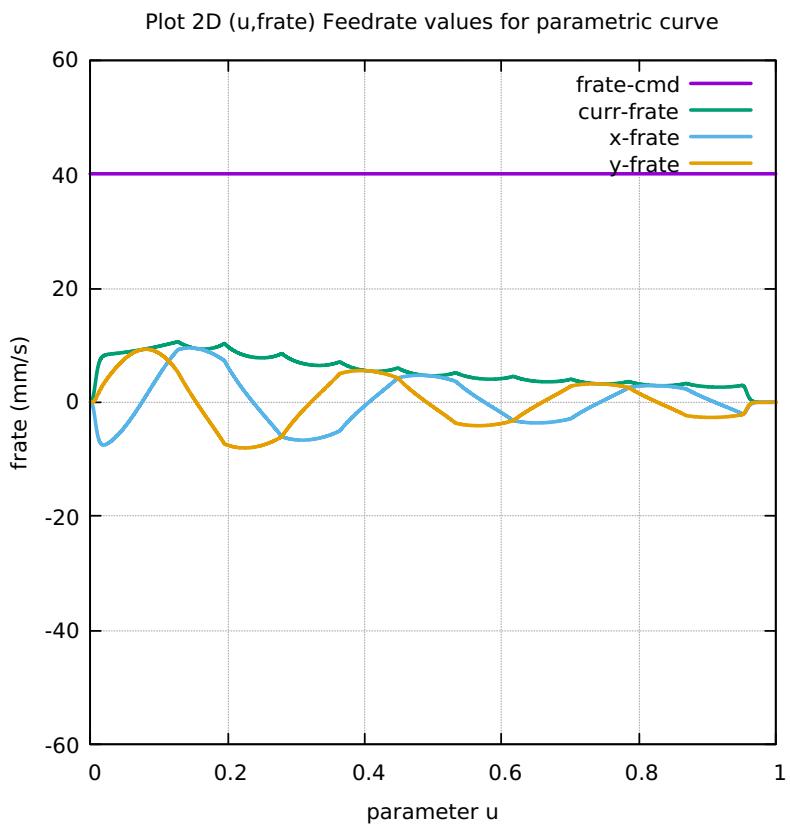


Figure 185: Snailshell FC10 Four Components FeedrateLimit

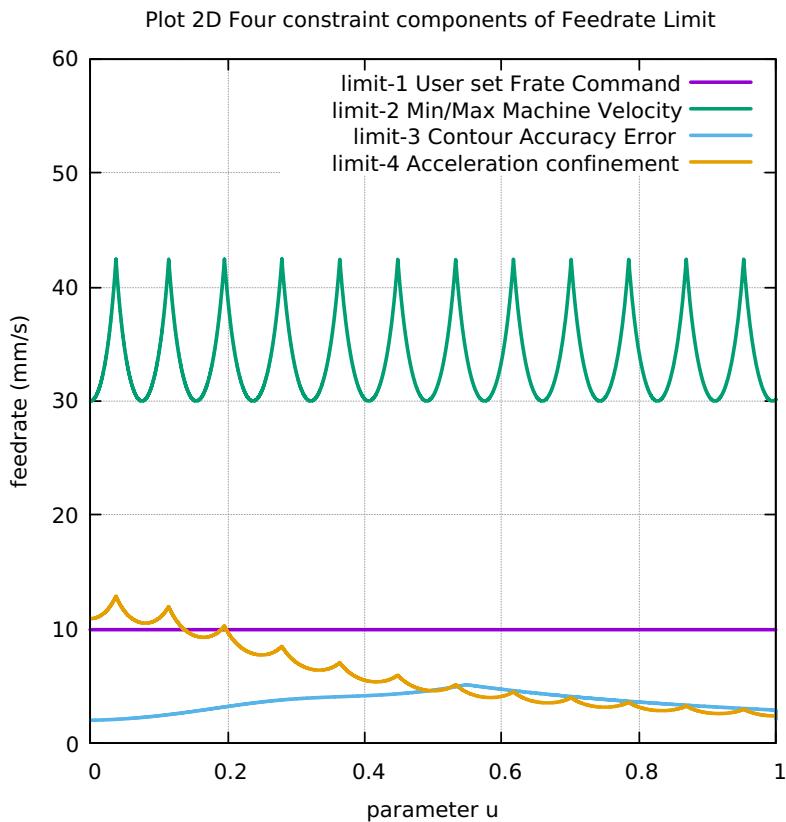


Figure 186: Snailshell FC20 Four Components FeedrateLimit

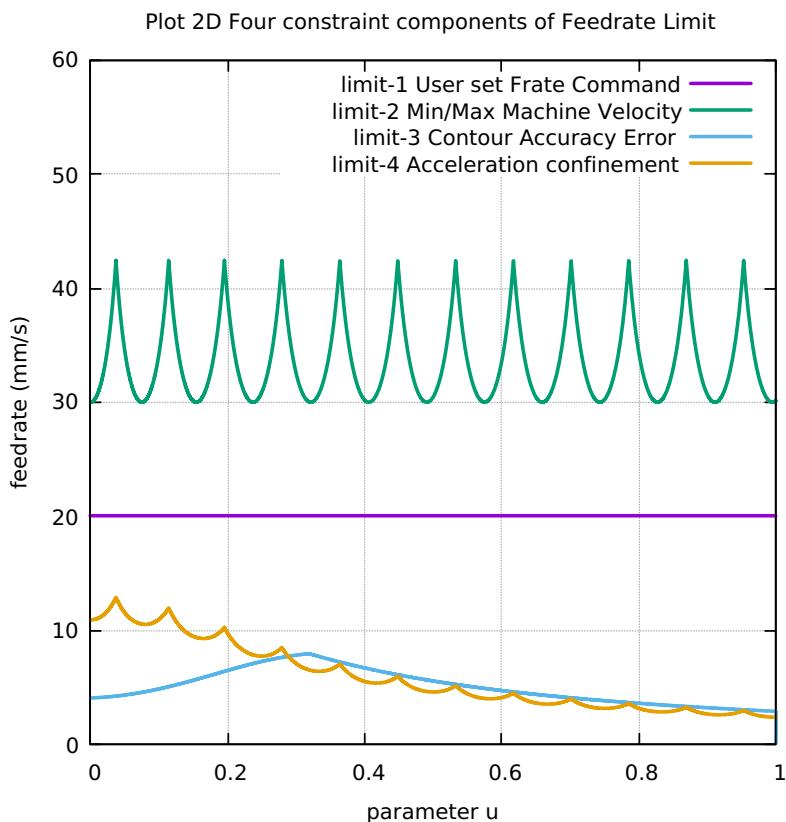


Figure 187: Snailshell FC30 Four Components FeedrateLimit

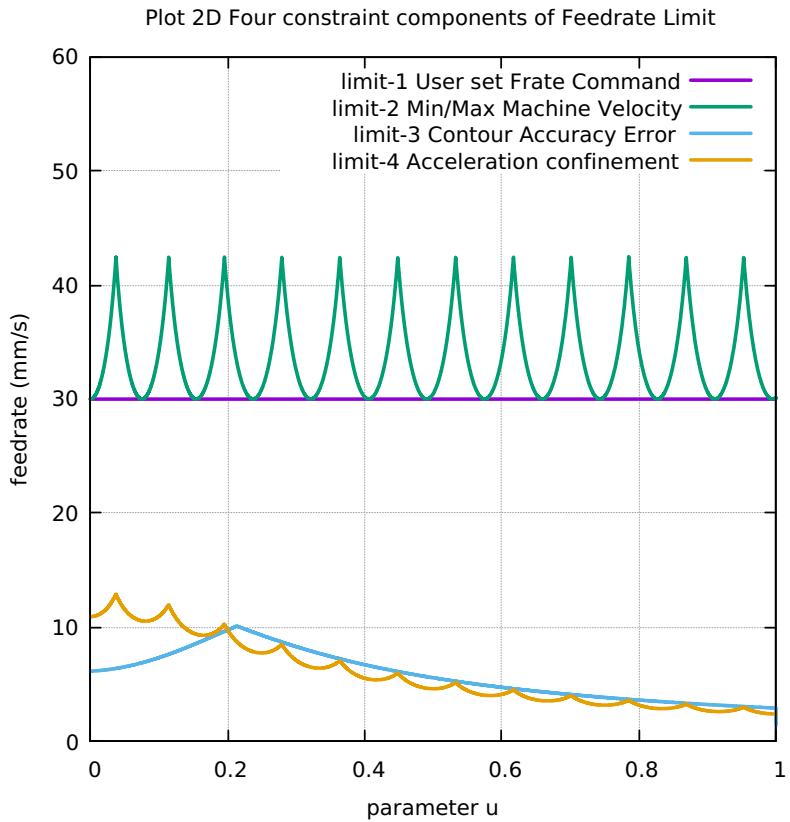


Figure 188: Snailshell FC40 Four Components FeedrateLimit

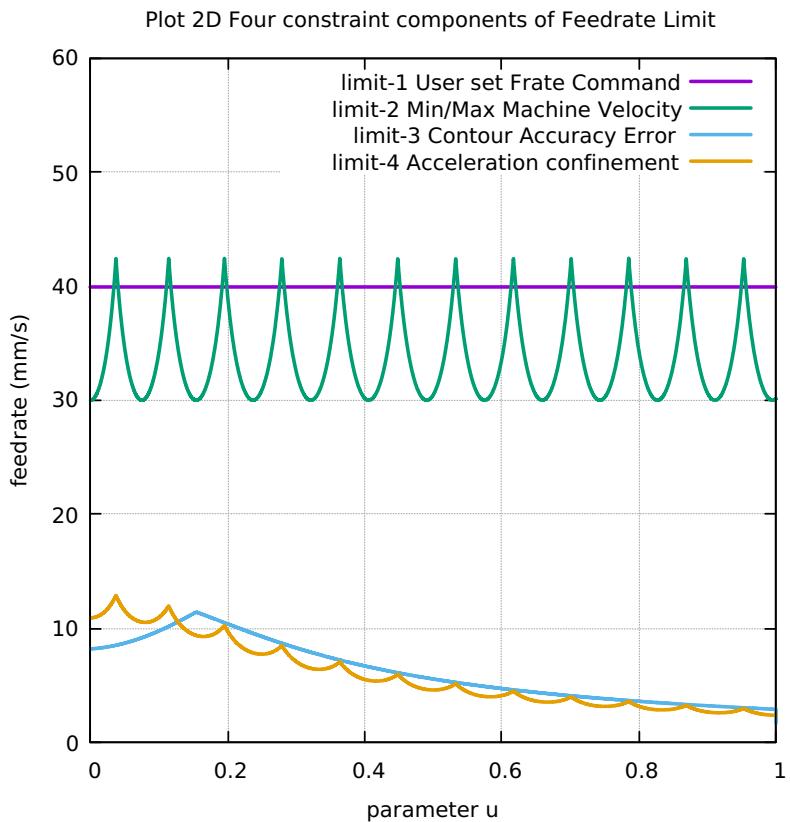


Figure 189: Snailshell Histogram Points FC10 FC20 FC30 FC40

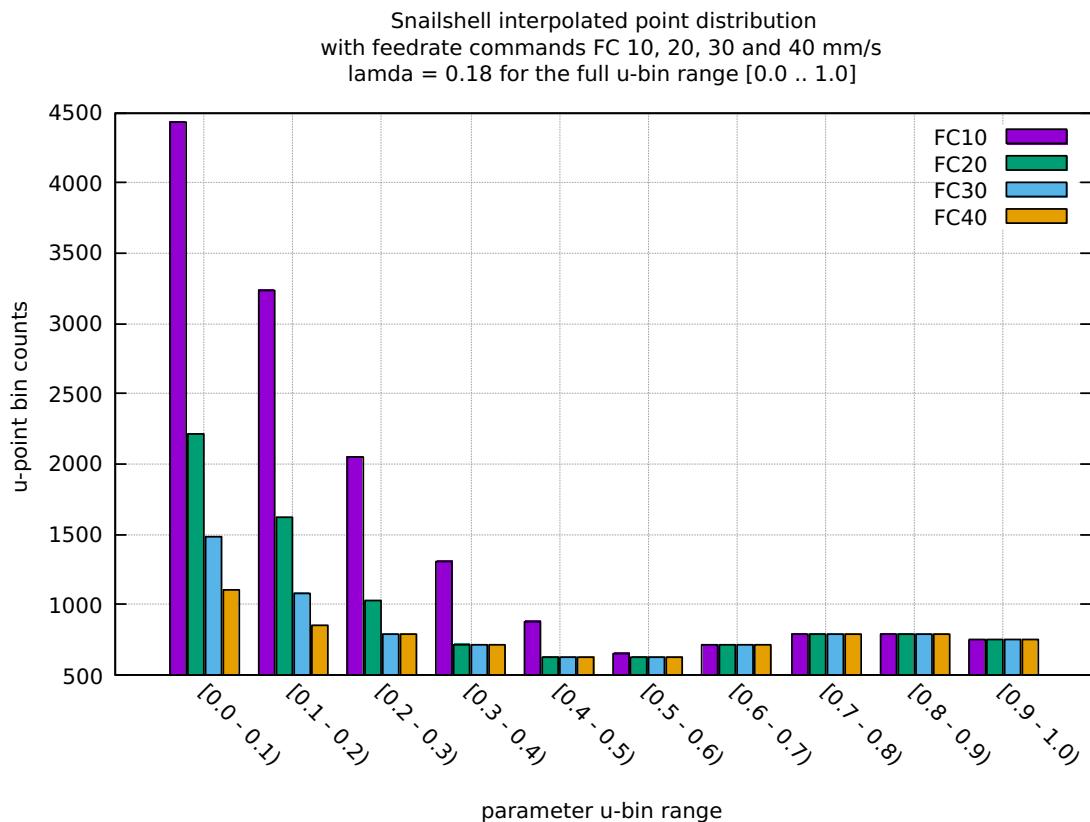


Table 10: Snailshell Table distribution of interpolated points

BINS	FC10	FC20	FC30	FC40
0.0 - 0.1	4435	2218	1479	1109
0.1 - 0.2	3237	1619	1080	849
0.2 - 0.3	2054	1028	796	793
0.3 - 0.4	1312	714	711	711
0.4 - 0.5	881	629	629	629
0.5 - 0.6	657	628	629	628
0.6 - 0.7	710	711	710	711
0.7 - 0.8	794	794	794	794
0.8 - 0.9	791	791	792	792
0.9 - 1.0	750	751	750	750
Tot Counts	15621	9883	8370	7766

Table 11: Snailshell Table FC10-20-30-40 Run Performance data

1	Curve Type	SNAIL SHELL	SNAIL SHELL	SNAIL SHELL
2	User Feedrate Command FC(mm/s)	FC10	FC20	FC40
3	User Lamda Acceleration Safety Factor	0.18	0.18	0.18
4	Total Interpolated Points (TIP)	15621	9883	8370
5	Total Sum-Chord-Error (SCE) (mm)	5.115139395656E-03	6.269762299809E-03	6.764496555494E-03
6	Ratio 1 = (SCE/TIP) = Chord-Error/Point	3.274737129101E-07	6.344628921078E-07	8.082801476275E-07
7	Total Sum-Arc-Length (SAL) (mm)	1.385725614787E+02	1.385823558880E+02	1.385855208343E+02
8	Total Sum-Chord-Length (SCL) (mm)	1.385595406106E+02	1.385613905727E+02	1.385601641834E+02
9	Difference = (SAL - SCL) (mm)	1.302086811782E-02	2.096531529295E-02	2.535665098060E-02
10	Percentage Difference = (SAL - SCL)/SAL	9.396425943836E-03	1.512841599395E-02	1.829675338949E-02
11	Ratio 2 = (SCE/SCL) = Chord Error/Chord-Length	3.691654413053E-05	4.524898511695E-05	4.881992306637E-05
12	Total Sum-Arc-Theta (SAT) (rad)	1.894562150080E+01	1.8945335923343E+01	1.894325318416E+01
13	Total Sum-Arc-Area (SAA) (mm2)	1.070830851582E-04	3.068808285119E-04	5.164360050036E-04
14	Ratio 3 = (SAA/SCL) = Arc-Area/Chord-Length	3.691654413053E-05	4.524898511695E-05	4.881992306637E-05
15	Average-Chord-Error (ACE) (mm)	3.274737129101E-07	6.344628921078E-07	8.082801476275E-07
16	Average-Arc-Length (AAL) (mm)	8.871482809134E-03	1.402371543089E-02	1.655938831812E-02
17	Average-Chord-Length (ACL) (mm)	8.870649206822E-03	1.402159386488E-02	1.655635848768E-02
18	Average-Arc-Theta (AAT) (rad)	1.212907906581E-03	1.917156033539E-03	2.263502591010E-03
19	Average-Arc-Area (AAA) (mm2)	6.855511213715E-09	3.105452626107E-08	6.170820946393E-08
20	Algorithm actual runtime on computer (ART) (s)	17.308275283	24.17500989	29.295160546
				32.06253681

.8 APPENDIX SKEWED-ASTROID CURVE

- .8.1 Plot of SkwAstroid curve [190]**
- .8.2 SkwAstroid Radius of Curvature [191]**
- .8.3 SkwAstroid Validation in LinuxCNC [192]**
- .8.4 SkwAstroid Direction of Travel 3D [193]**
- .8.5 SkwAstroid First and Second Order Taylors App [194]**
- .8.6 SkwAstroid First minus Second Order Taylors App [195]**
- .8.7 SkwAstroid Separate First Second Order Taylors App [196]**
- .8.8 SkwAstroid Separation SAL and SCL [197]**
- .8.9 SkwAstroid Chord-error in close view 2 scales [198]**
- .8.10 SkwAstroid Four Components Feedrate Limit [199]**
- .8.11 SkwAstroid FrateCommand FrateLimit and Curr-Frate [200]**
- .8.12 SkwAstroid FeedRateLimit minus CurrFeedRate [201]**
- .8.13 SkwAstroid FC20-Nominal X and Y Feedrate Profiles [202]**
- .8.14 SkwAstroid FC20 Nominal Tangential Acceleration [203]**
- .8.15 SkwAstroid FC20 Nominal Rising S-Curve Profile [204]**
- .8.16 SkwAstroid FC20 Nominal Falling S-Curve Profile [205]**
- .8.17 SkwAstroid FC10 Colored Feedrate Profile ngcode [206]**
- .8.18 SkwAstroid FC20 Colored Feedrate Profile ngcode [207]**

- .8.19 SkwAstroid FC30 Colored Feedrate Profile ngcode [208]
- .8.20 SkwAstroid FC40 Colored Feedrate Profile ngcode [209]
- .8.21 SkwAstroid FC10 Tangential Acceleration [210]
- .8.22 SkwAstroid FC20 Tangential Acceleration [211]
- .8.23 SkwAstroid FC30 Tangential Acceleration [212]
- .8.24 SkwAstroid FC40 Tangential Acceleration [213]
- .8.25 SkwAstroid FC20 Nominal Separation NAL and NCL [214]
- .8.26 SkwAstroid SAL minus SCL for FC10 FC20 FC30 FC40 [215]
- .8.27 SkwAstroid FC10 FrateCmd CurrFrate X-Frate Y-Frate [216]
- .8.28 SkwAstroid FC20 FrateCmd CurrFrate X-Frate Y-Frate [217]
- .8.29 SkwAstroid FC30 FrateCmd CurrFrate X-Frate Y-Frate [218]
- .8.30 SkwAstroid FC40 FrateCmd CurrFrate X-Frate Y-Frate [219]
- .8.31 SkwAstroid FC10 Four Components FeedrateLimit [220]
- .8.32 SkwAstroid FC20 Four Components FeedrateLimit [221]
- .8.33 SkwAstroid FC30 Four Components FeedrateLimit [222]
- .8.34 SkwAstroid FC40 Four Components FeedrateLimit [223]
- .8.35 SkwAstroid Histogram Points FC10 FC20 FC30 FC40 [224]
- .8.36 SkwAstroid Table distribution of interpolated points [12]
- .8.37 SkwAstroid Table FC10-20-30-40 Run Performance data [13]

Figure 190: Plot of SkwAstroid curve

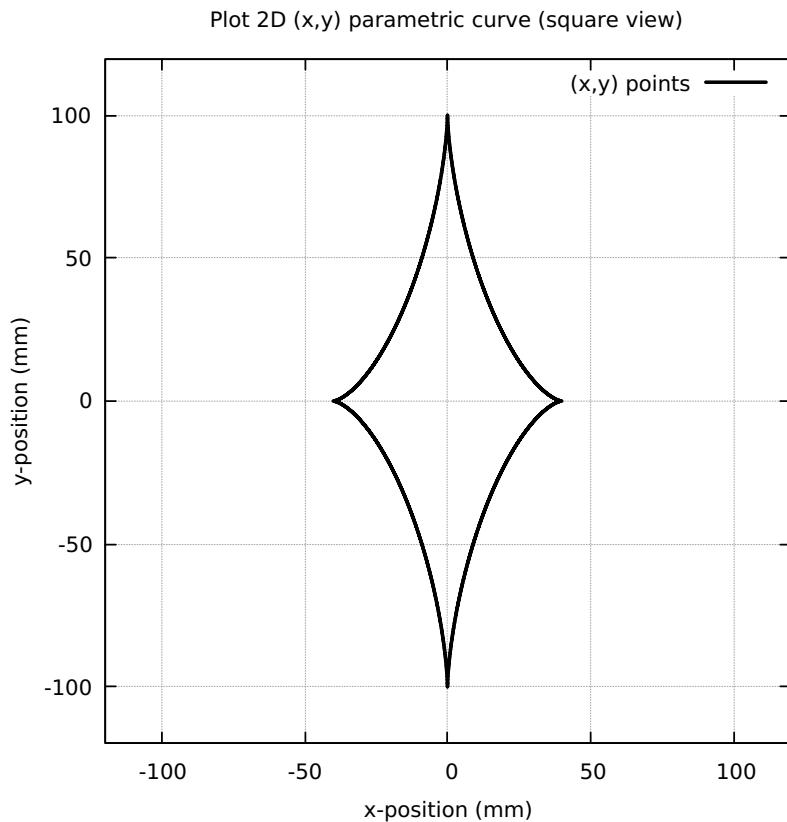


Figure 191: SkwAstroid Radius of Curvature

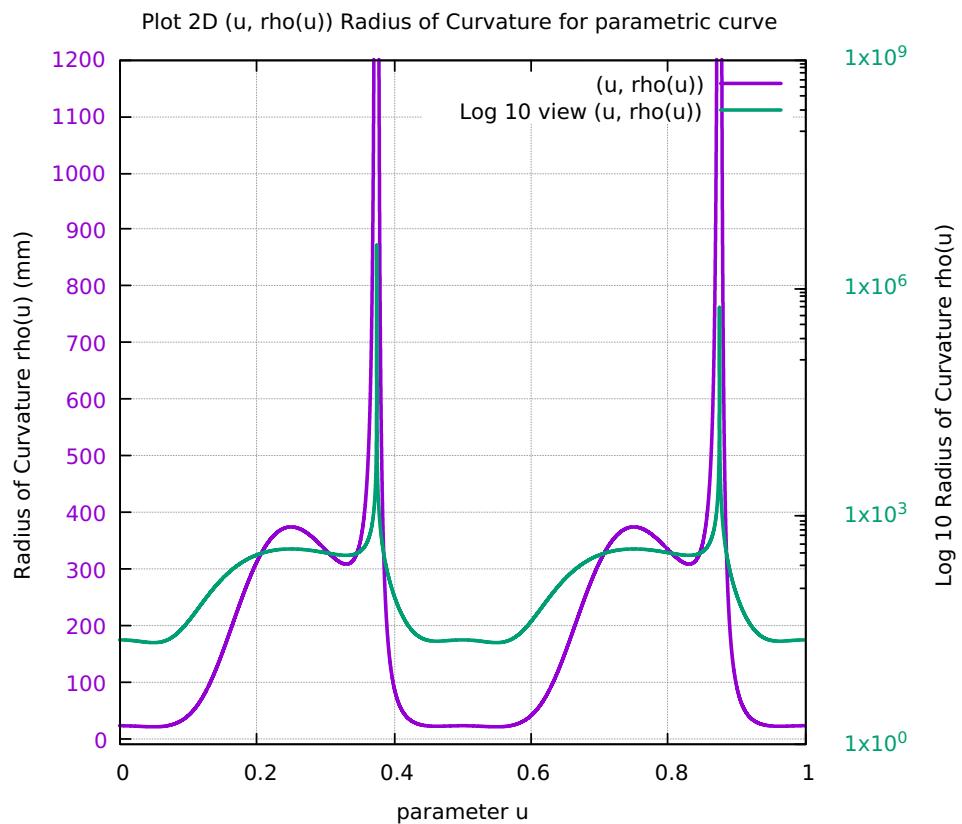


Figure 192: SkwAstroid Validation in LinuxCNC

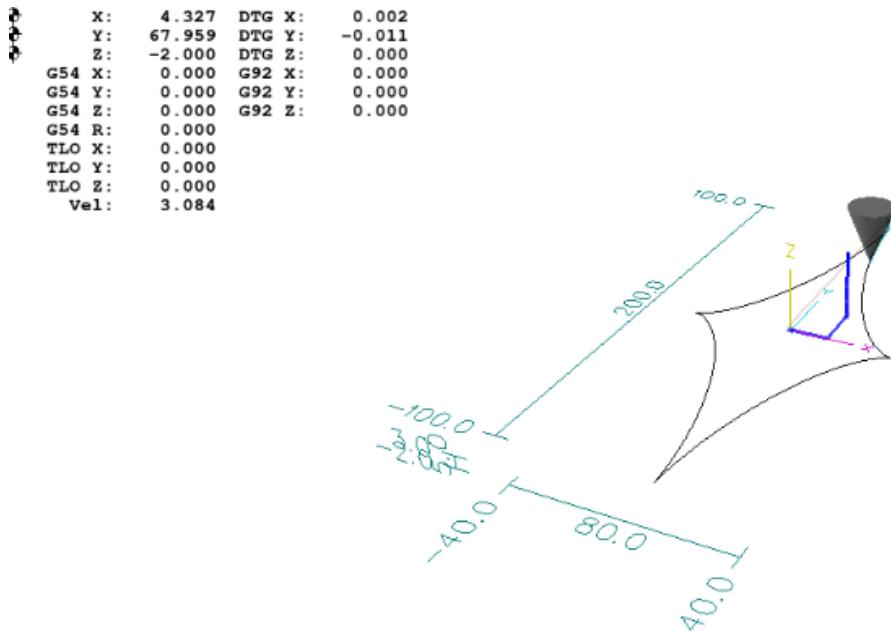


Figure 193: SkwAstroid Direction of Travel 3D

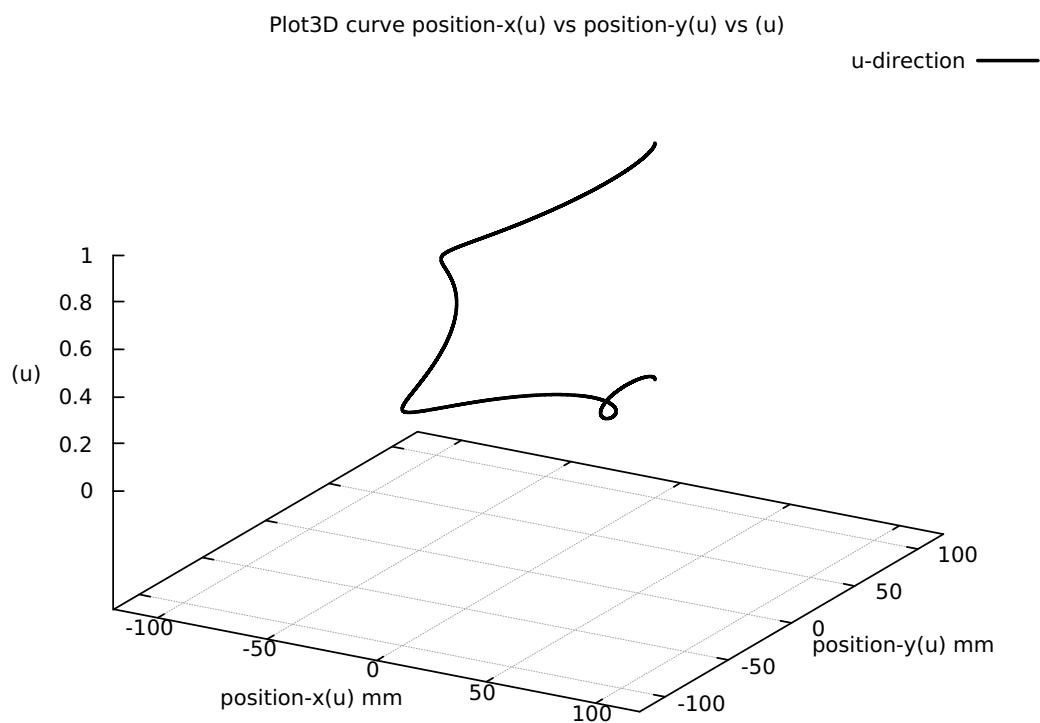


Figure 194: SkwAstroid First and Second Order Taylor's Approximation

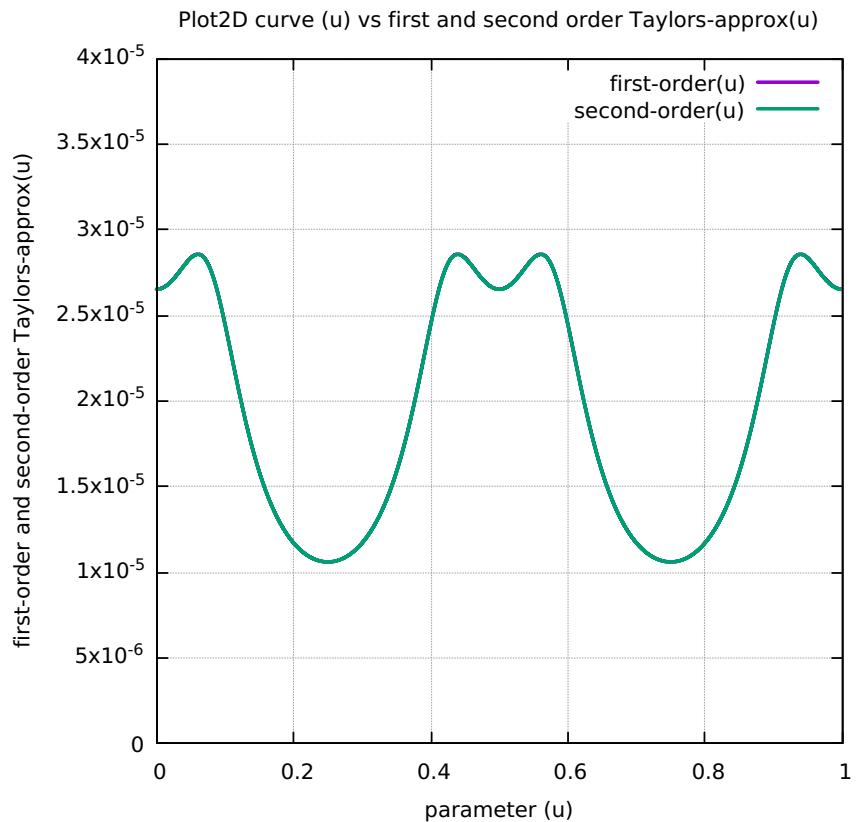


Figure 195: SkwAstroid First minus Second Order Taylor's Approximation

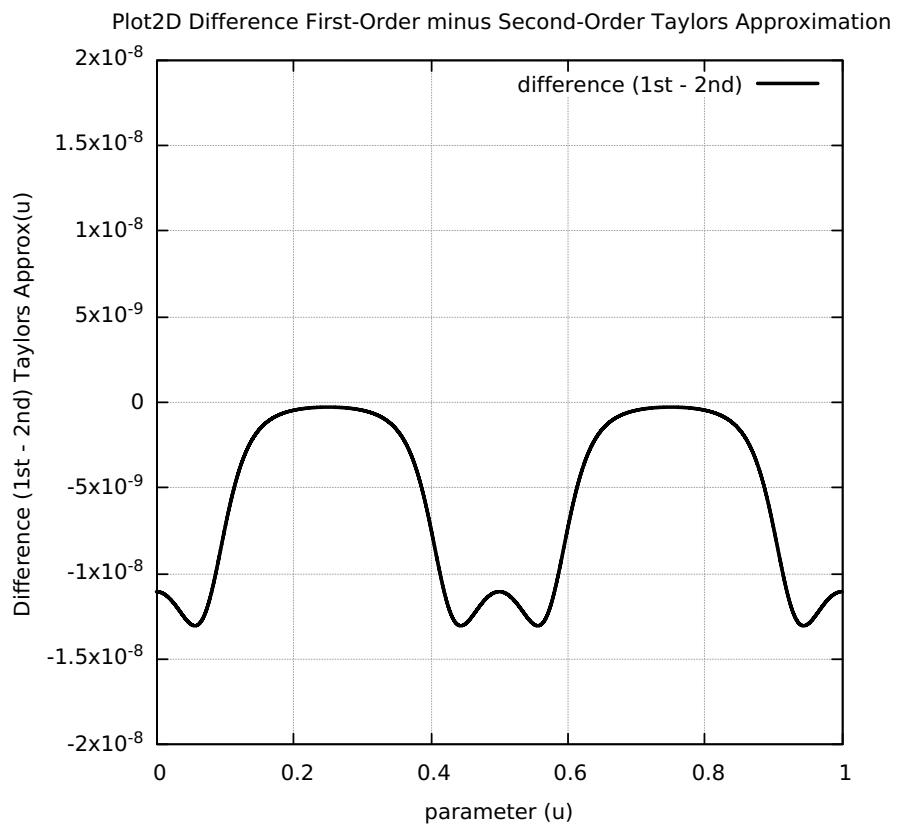


Figure 196: SkwAstroid Separation First and Second Order Taylor's Approximation

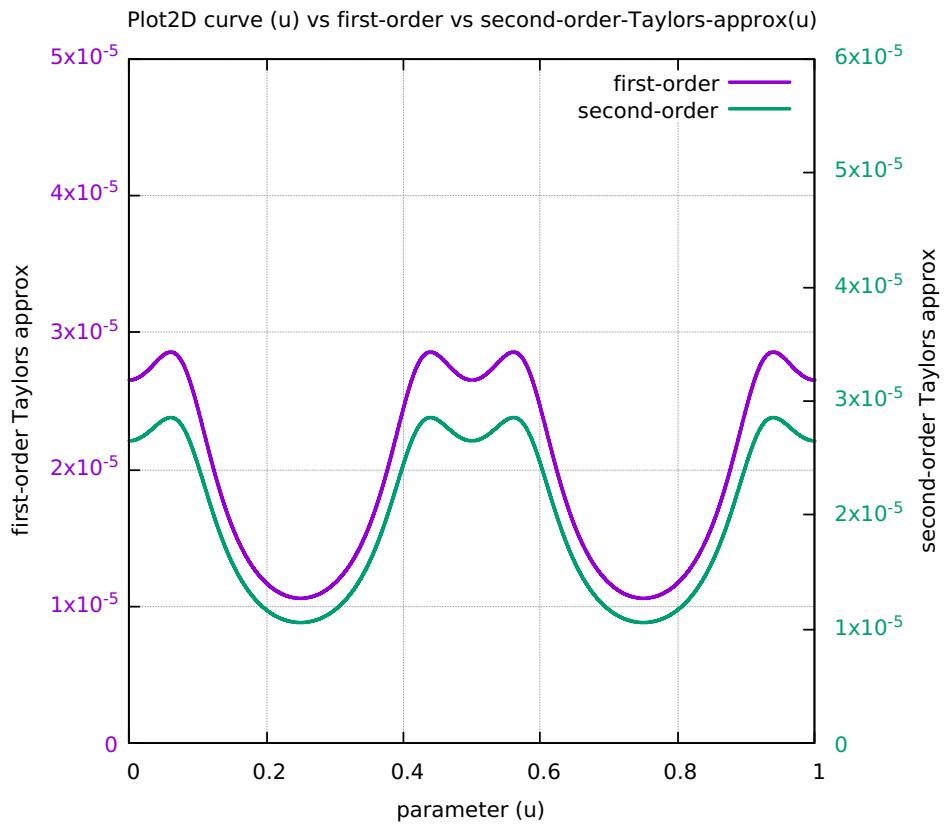


Figure 197: SkwAstroid Separation SAL and SCL

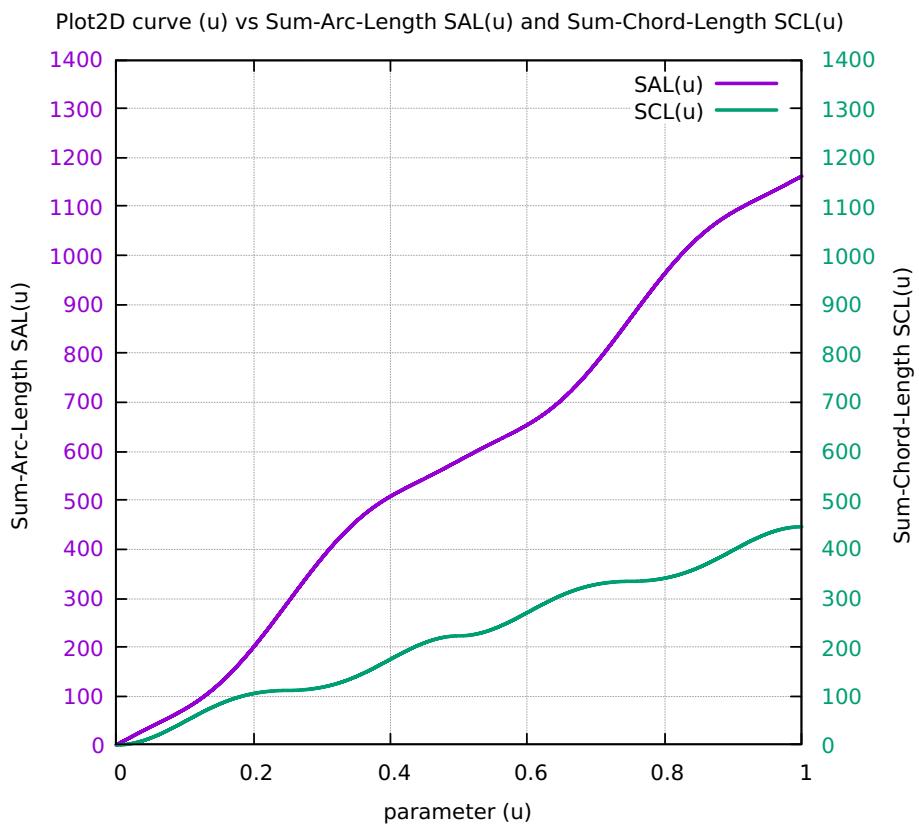


Figure 198: SkwAstroid Chord-error in close view 2 scales

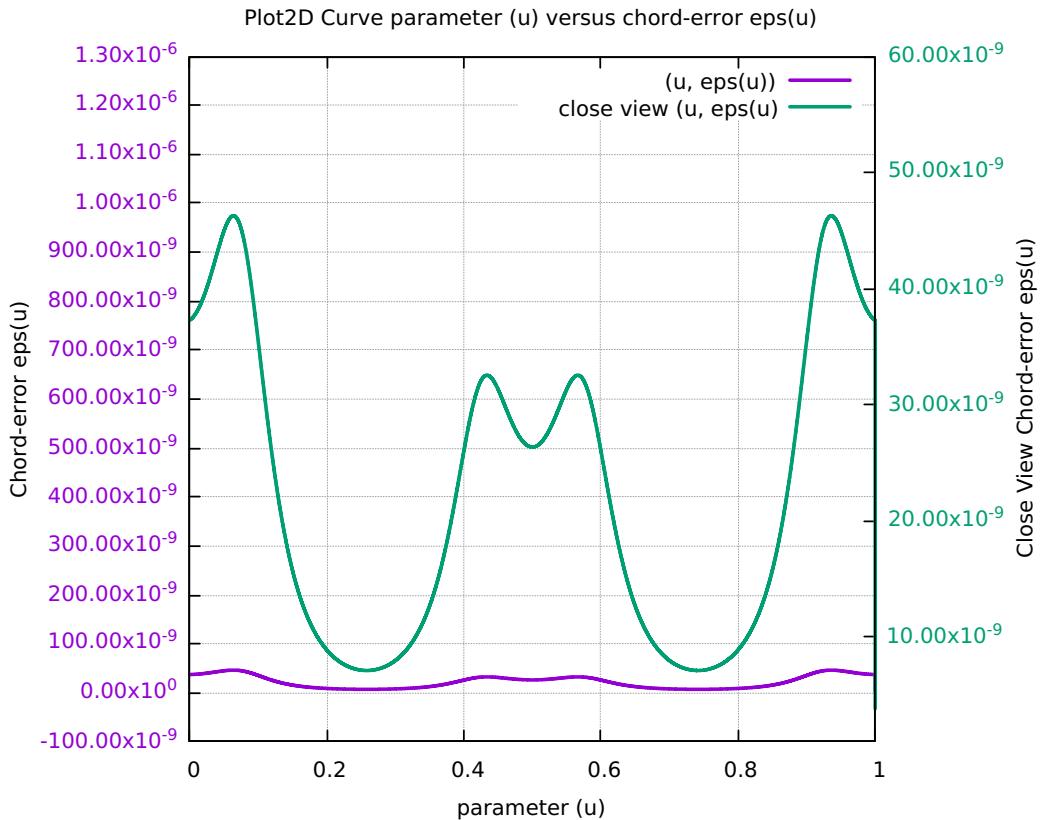


Figure 199: SkwAstroid Four Components Feedrate Limit

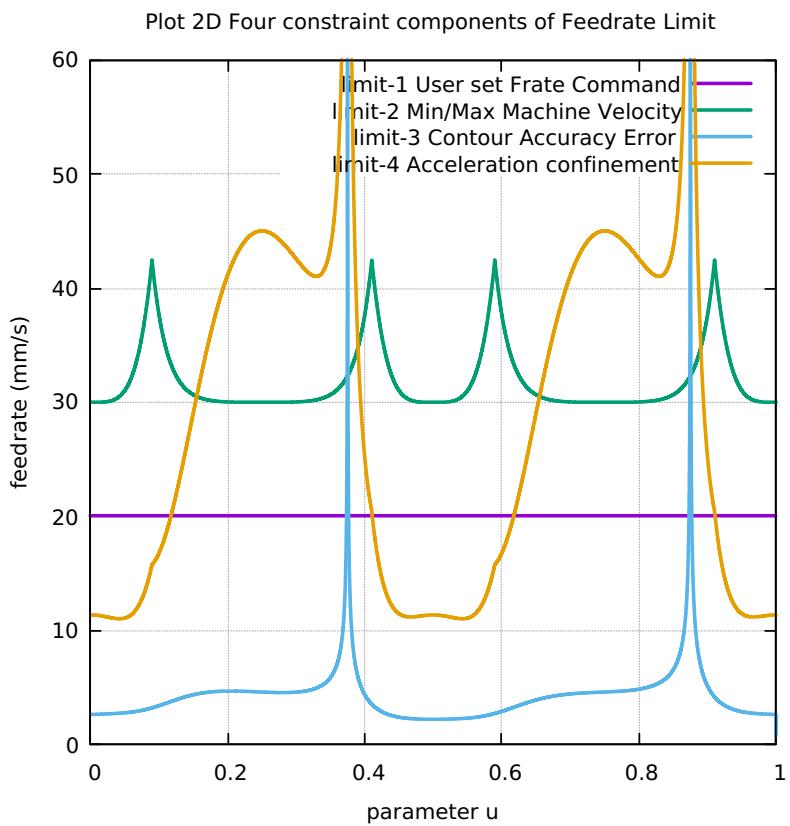


Figure 200: SkwAstroid FrateCommand FrateLimit and Curr-Frate

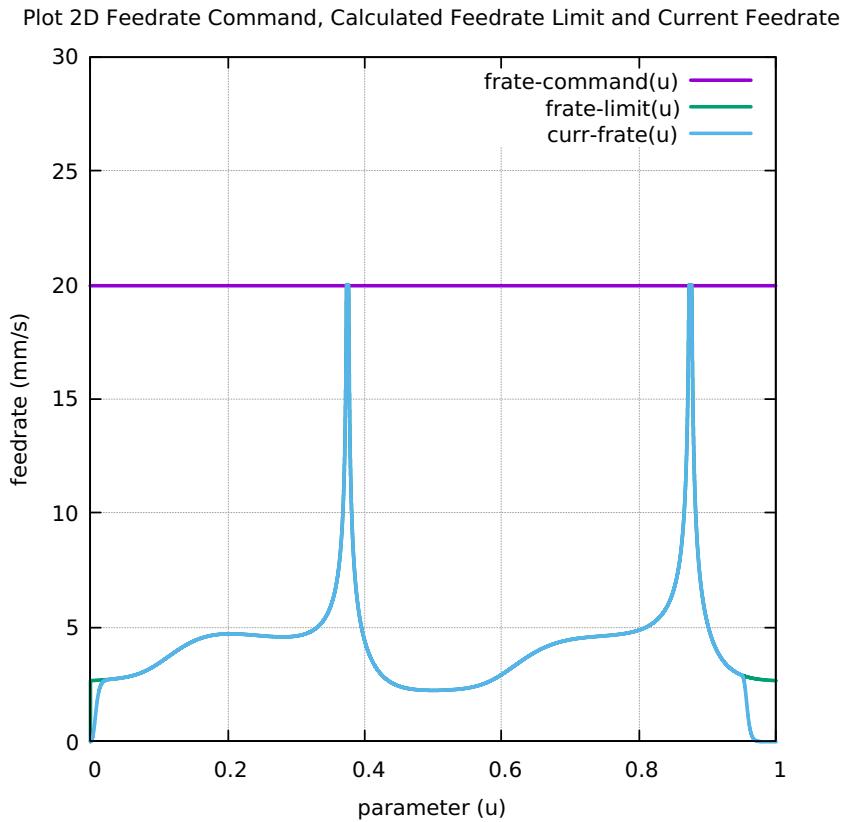


Figure 201: SkwAstroid FeedRateLimit minus CurrFeedRate

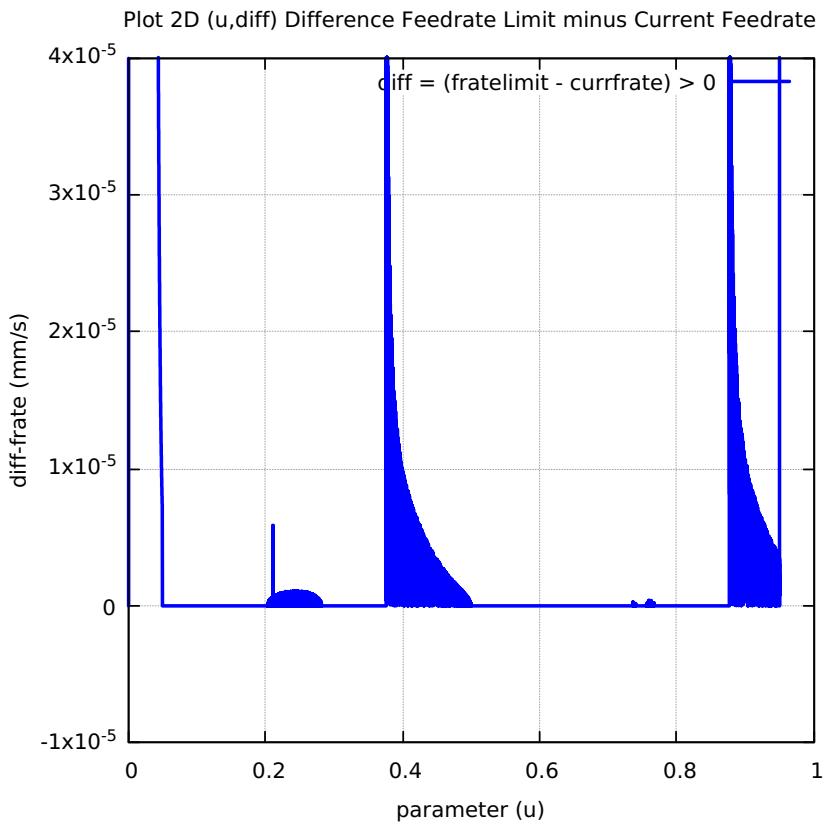


Figure 202: SkwAstroid FC20-Nominal X and Y Feedrate Profiles

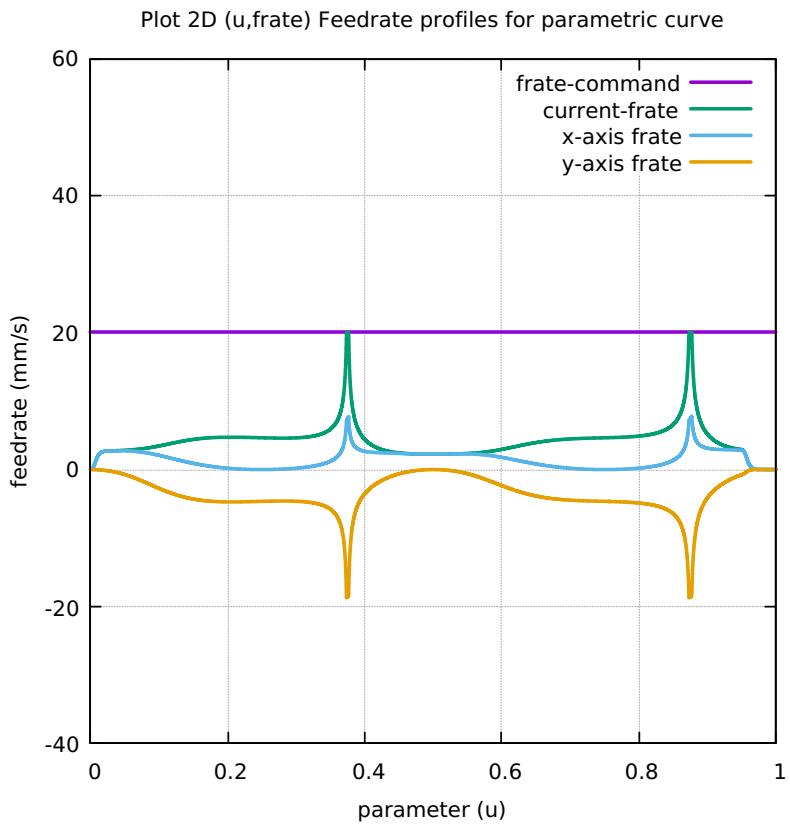


Figure 203: SkwAstroid FC20 Nominal Tangential Acceleration

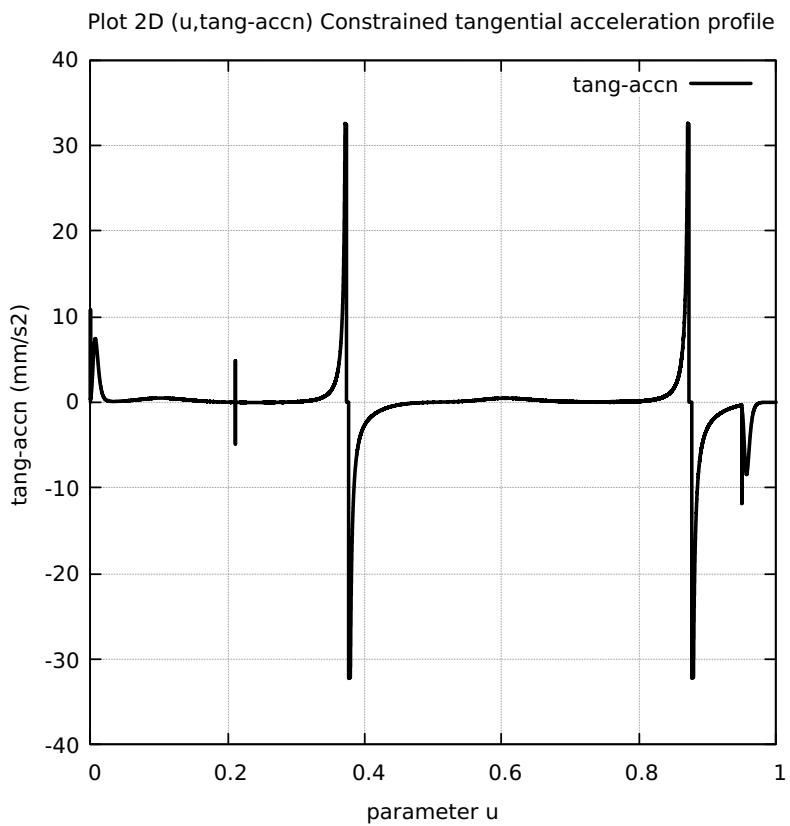


Figure 204: SkwAstroid FC20 Nominal Rising S-Curve Profile

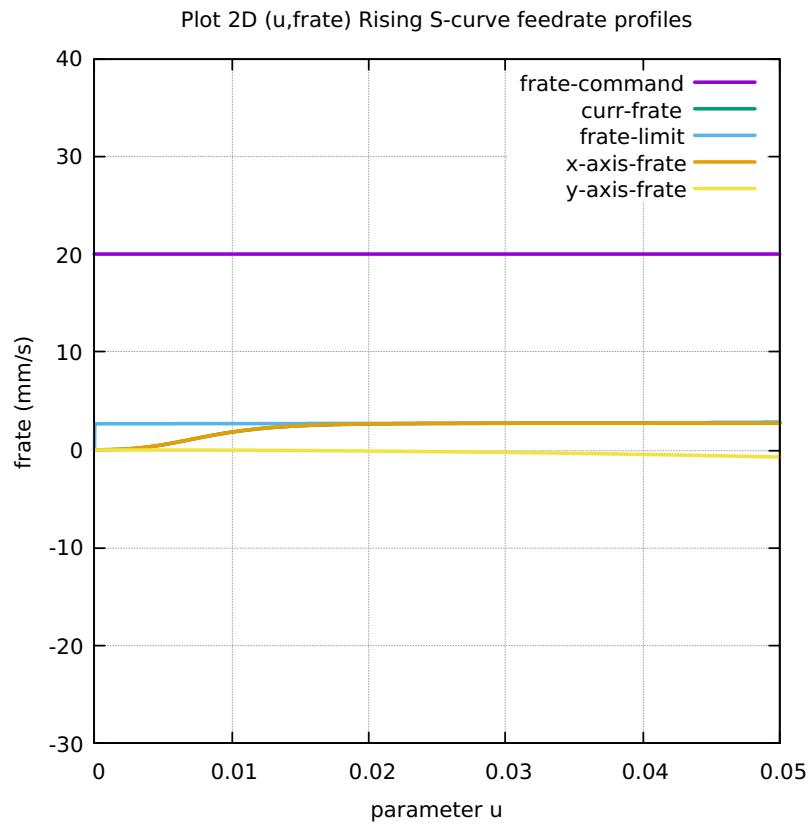


Figure 205: SkwAstroid FC20 Nominal Falling S-Curve Profile

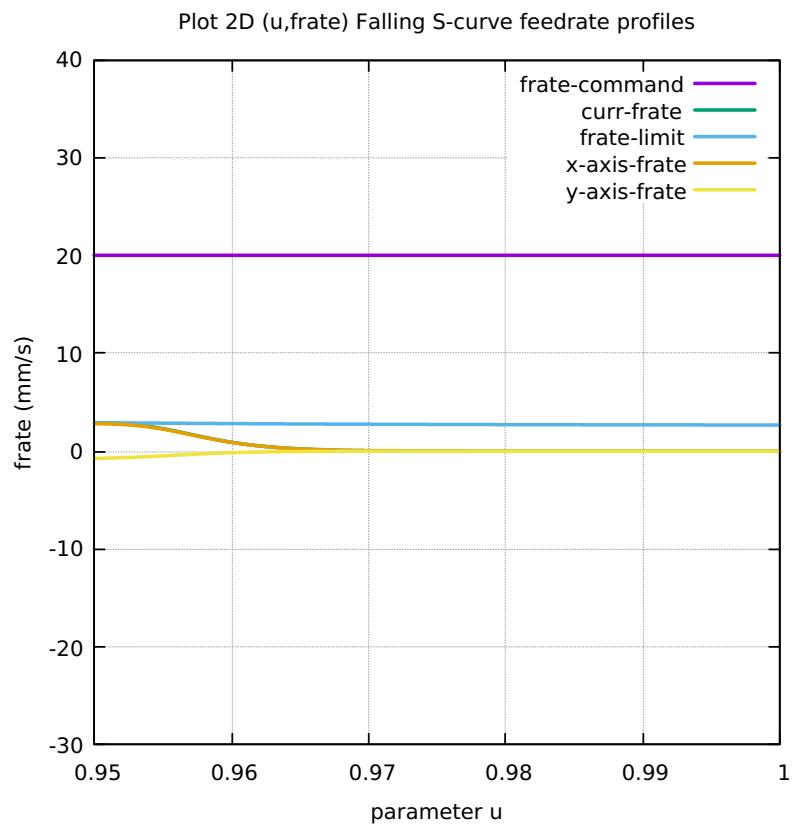


Figure 206: SkwAstroid FC10 Colored Feedrate Profile data ngcode

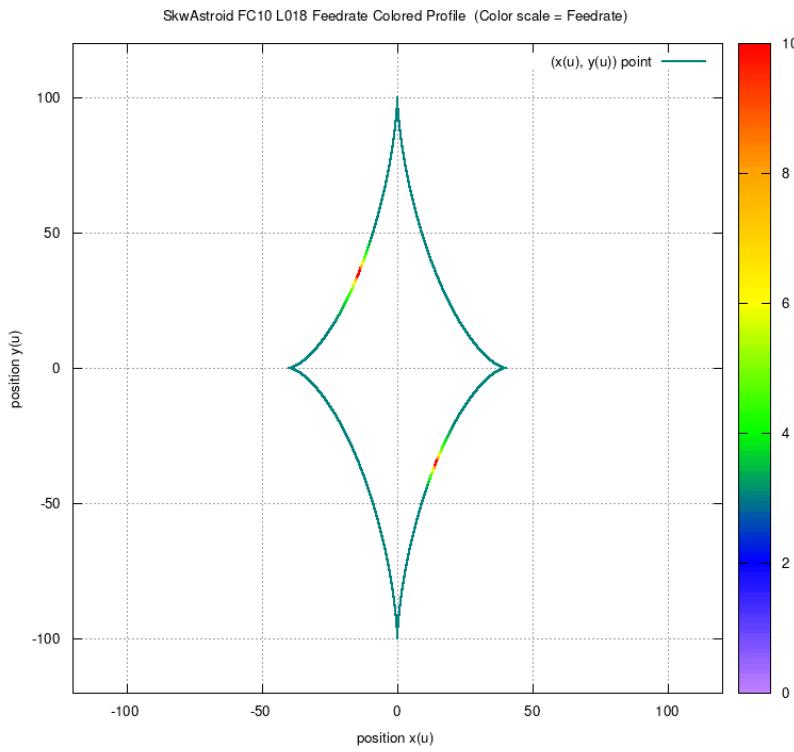


Figure 207: SkwAstroid FC20 Colored Feedrate Profile data ngcode

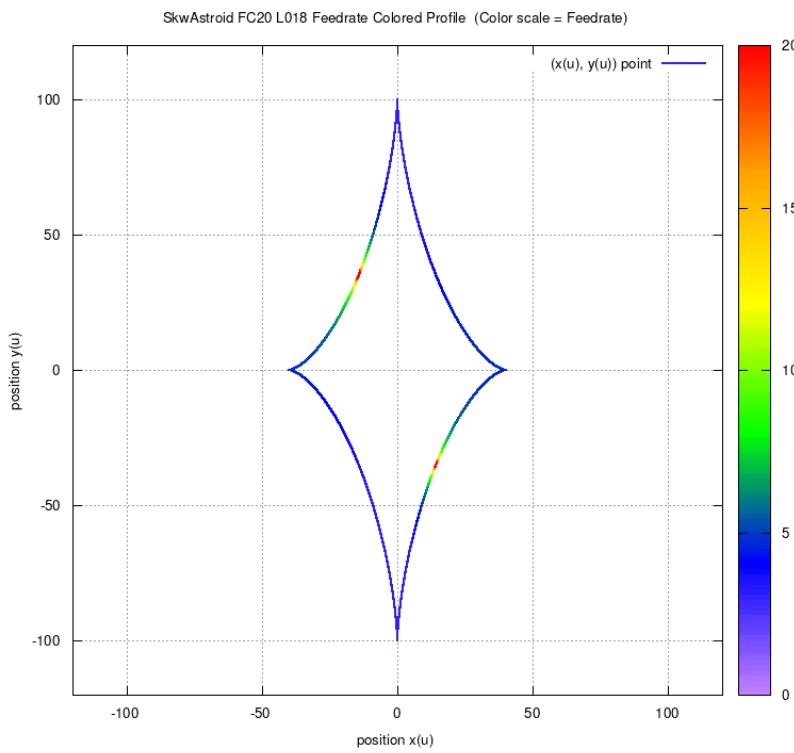


Figure 208: SkwAstroid FC30 Colored Feedrate Profile data ngcode

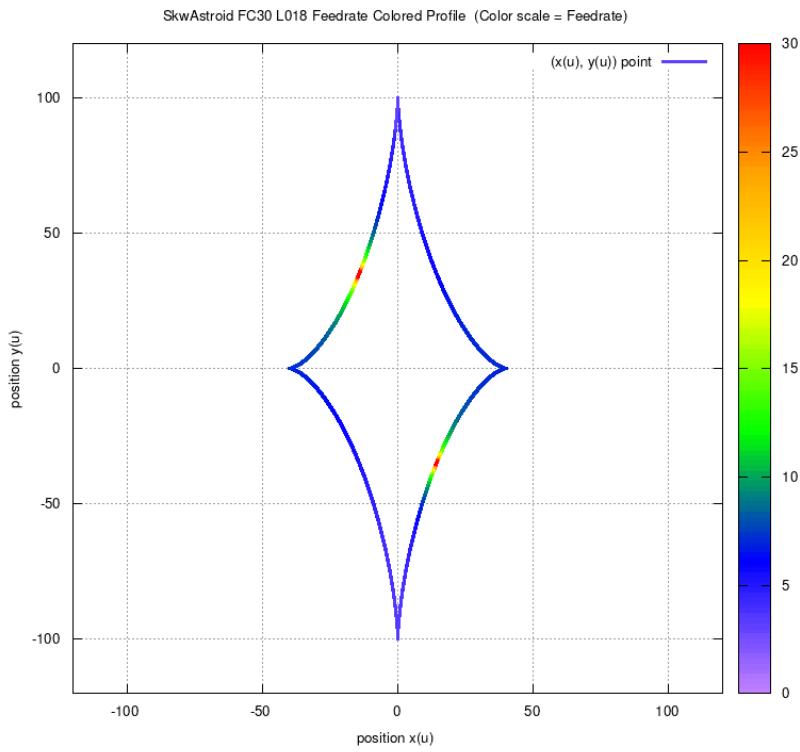


Figure 209: SkwAstroid FC40 Colored Feedrate Profile data ngcode

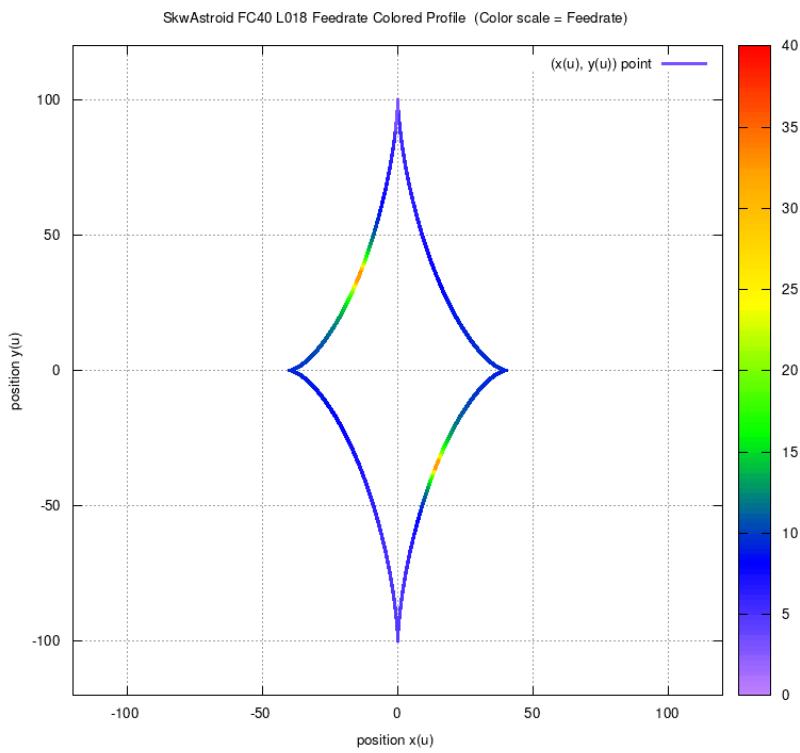


Figure 210: SkwAstroid FC10 Tangential Acceleration

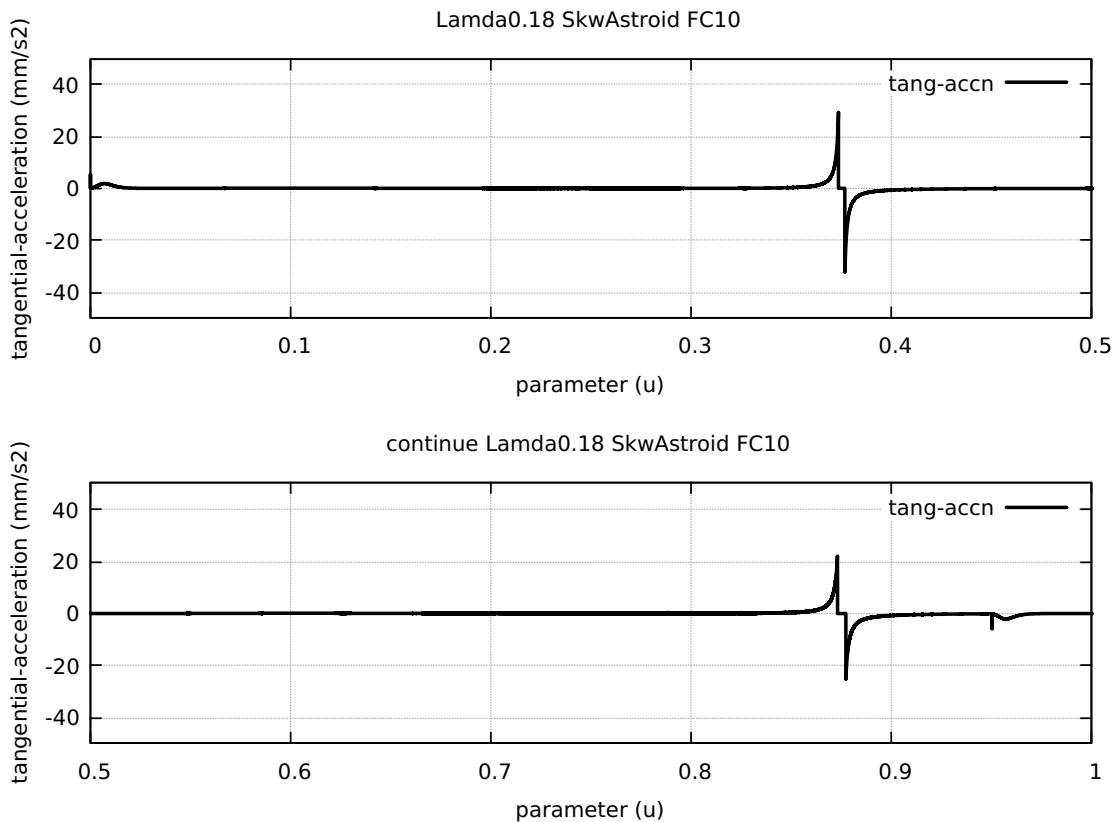


Figure 211: SkwAstroid FC20 Tangential Acceleration

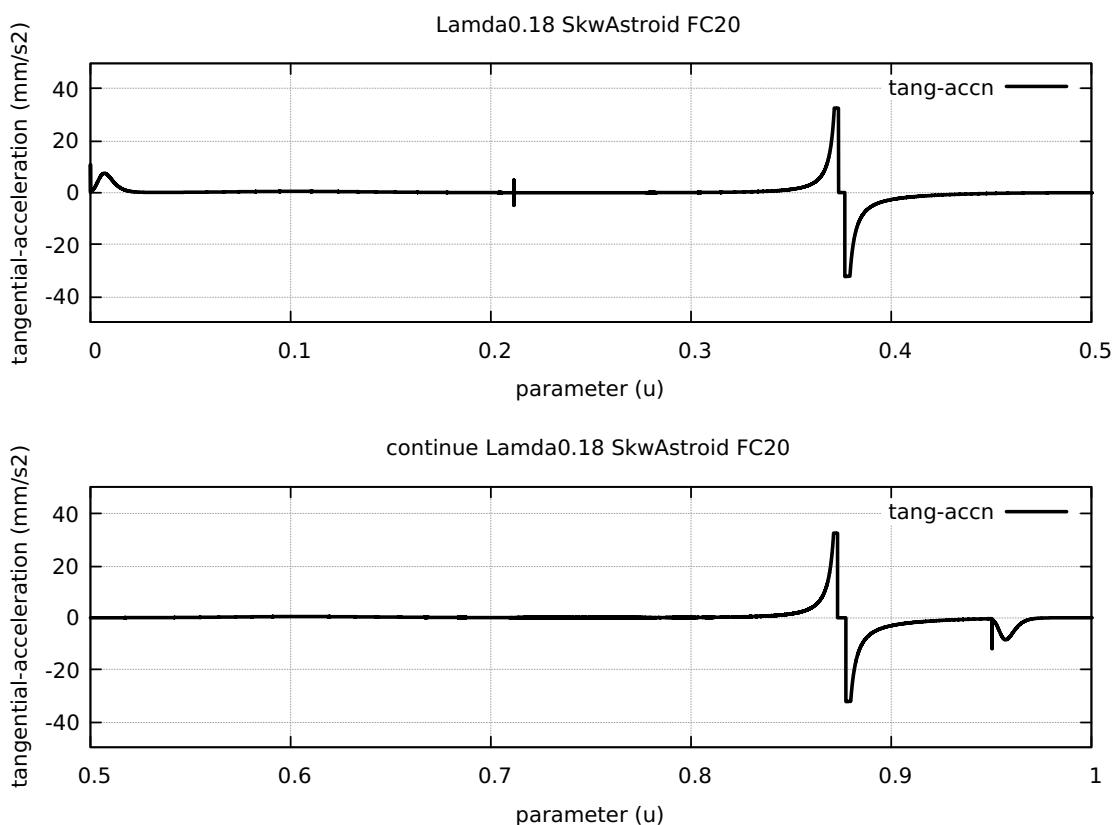


Figure 212: SkwAstroid FC30 Tangential Acceleration

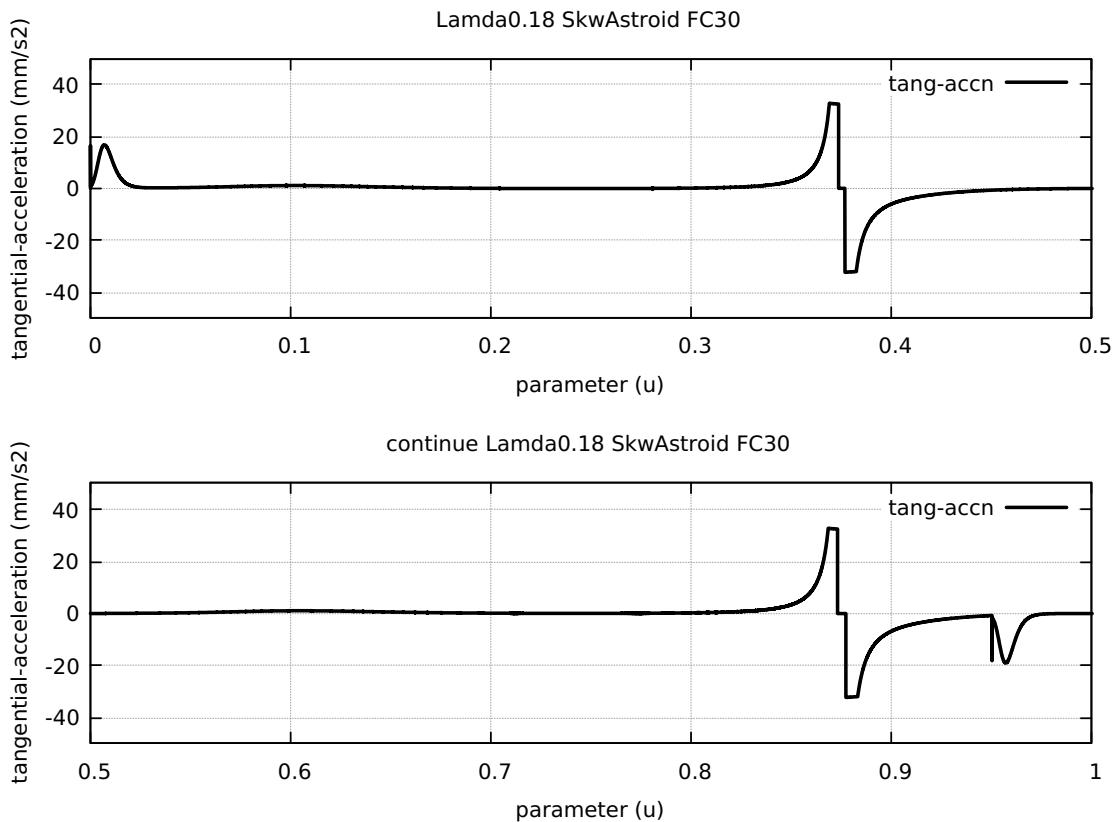


Figure 213: SkwAstroid FC40 Tangential Acceleration

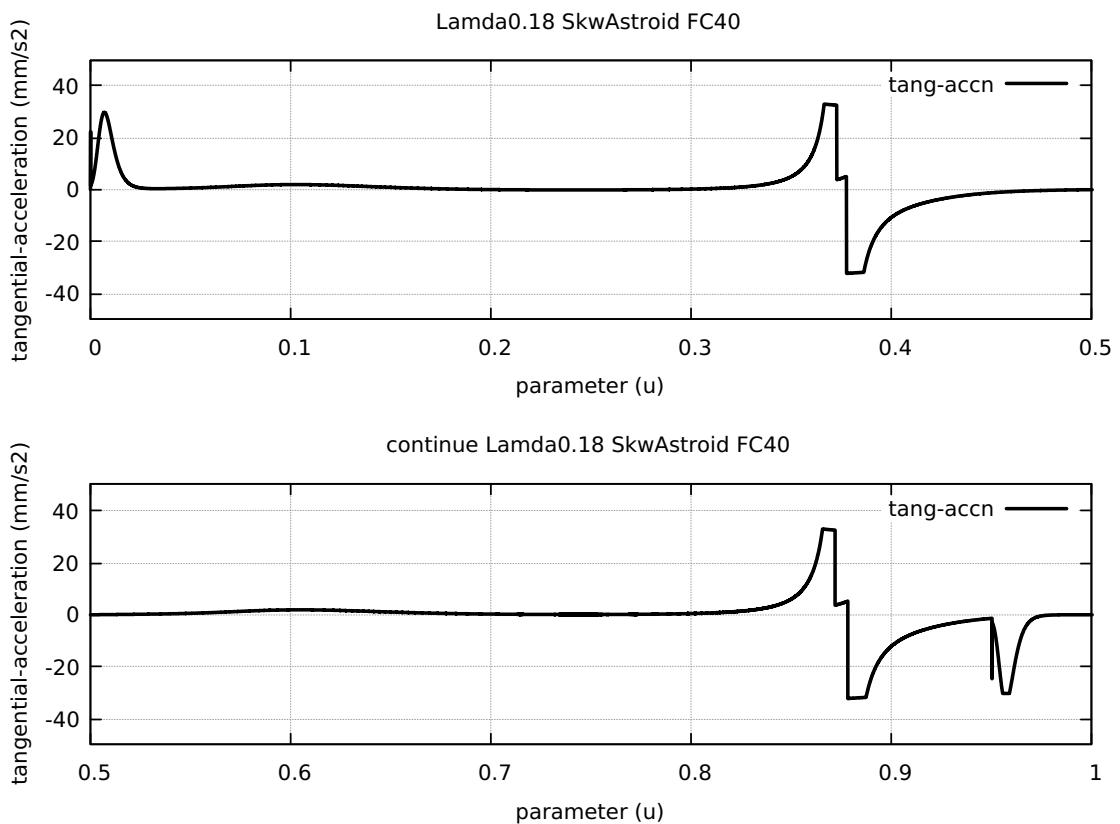


Figure 214: SkwAstroid FC20 Nominal Separation NAL and NCL

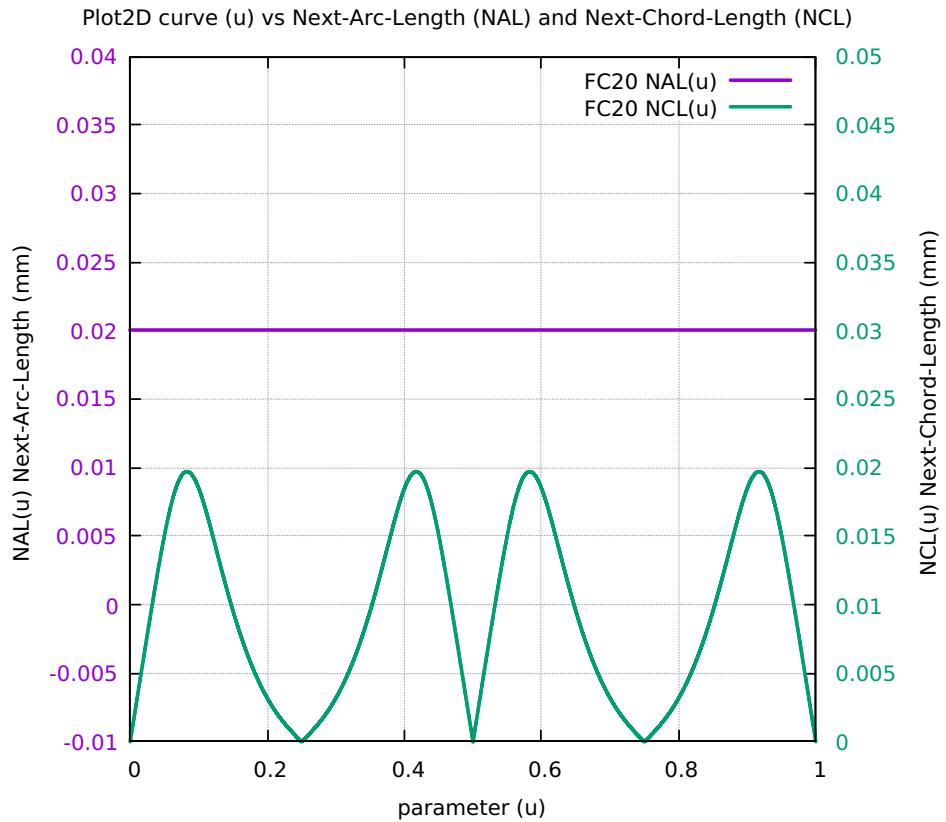


Figure 215: SkwAstroid Difference SAL minus SCL for FC10 FC20 FC30 FC40

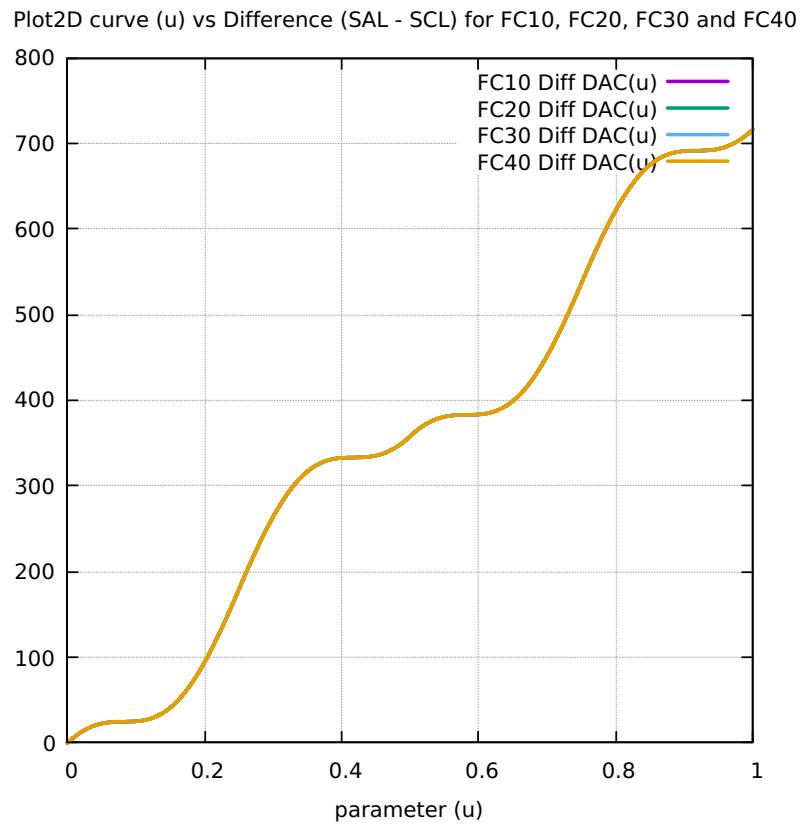


Figure 216: SkwAstroid FC10 FrateCmd CurrFrate X-Frate Y-Frate

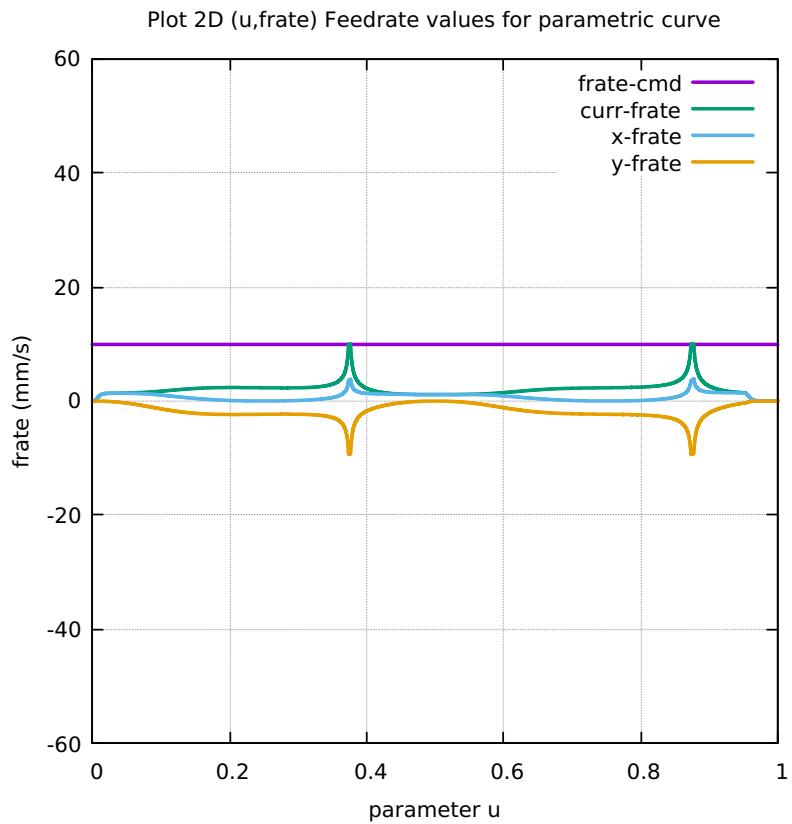


Figure 217: SkwAstroid FC20 FrateCmd CurrFrate X-Frate Y-Frate

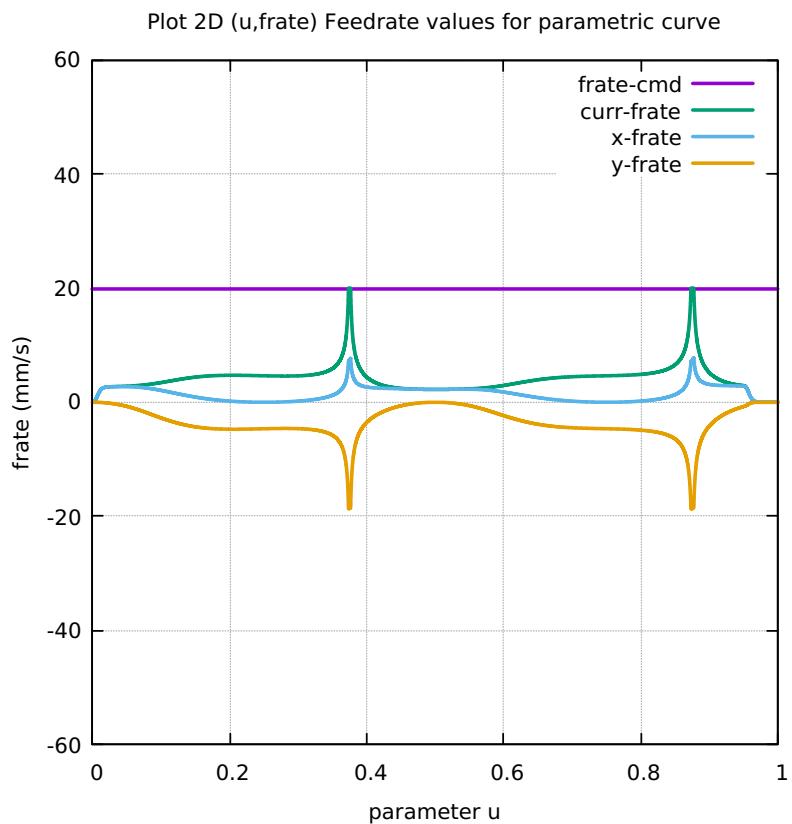


Figure 218: SkwAstroid FC30 FrateCmd CurrFrate X-Frate Y-Frate

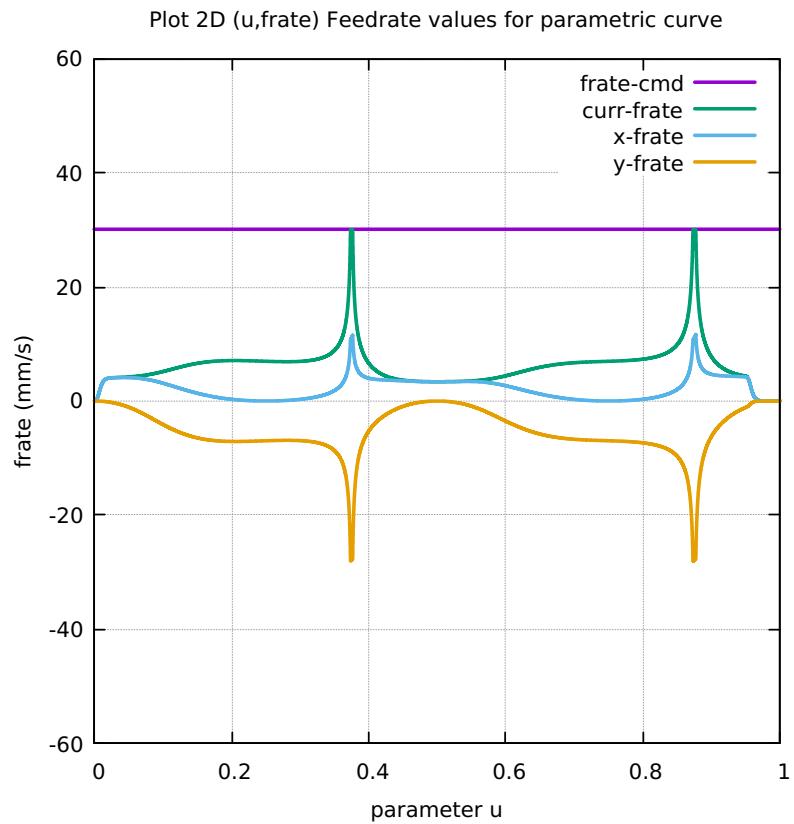


Figure 219: SkwAstroid FC40 FrateCmd CurrFrate X-Frate Y-Frate

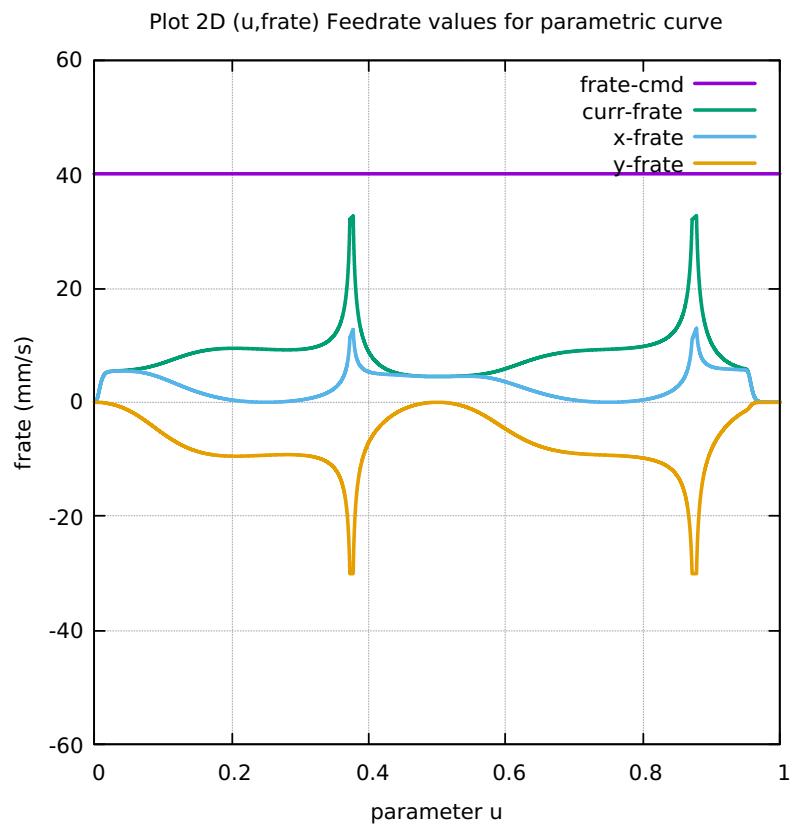


Figure 220: SkwAstroid FC10 Four Components FeedrateLimit

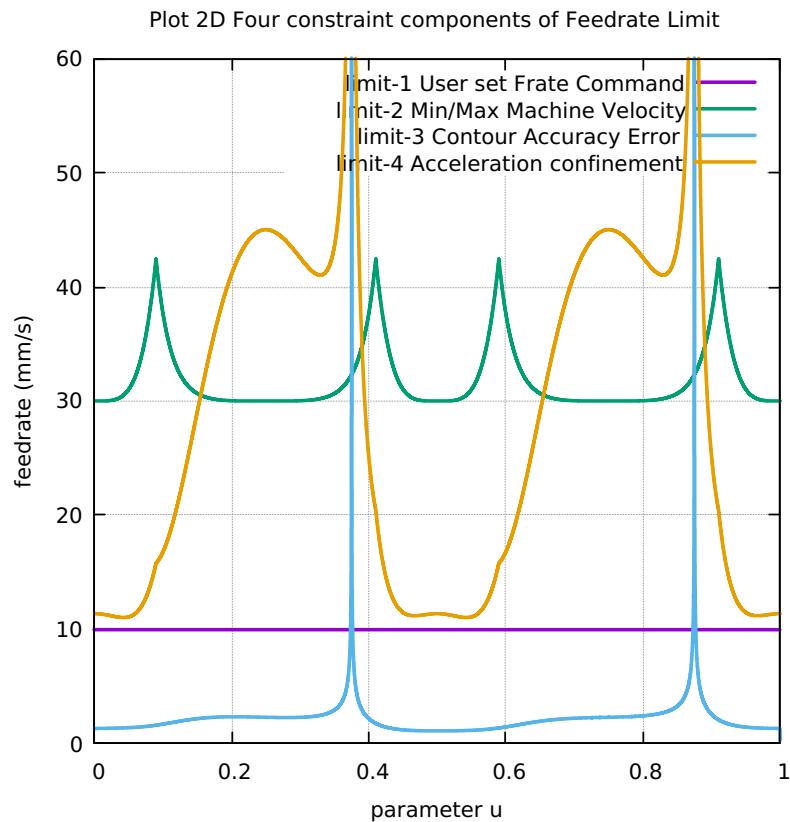


Figure 221: SkwAstroid FC20 Four Components FeedrateLimit

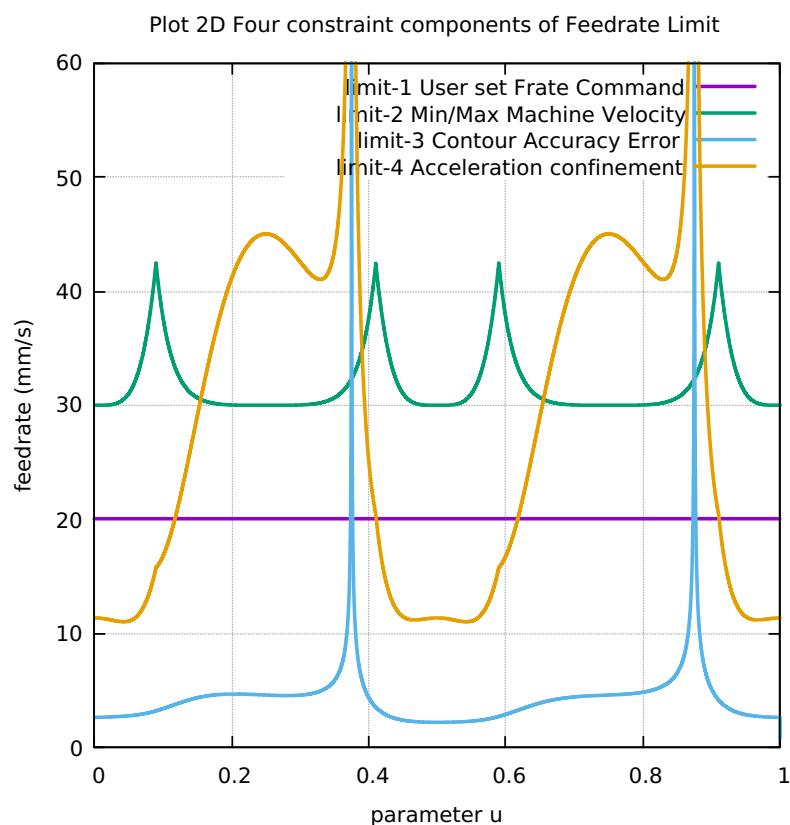


Figure 222: SkwAstroid FC30 Four Components FeedrateLimit

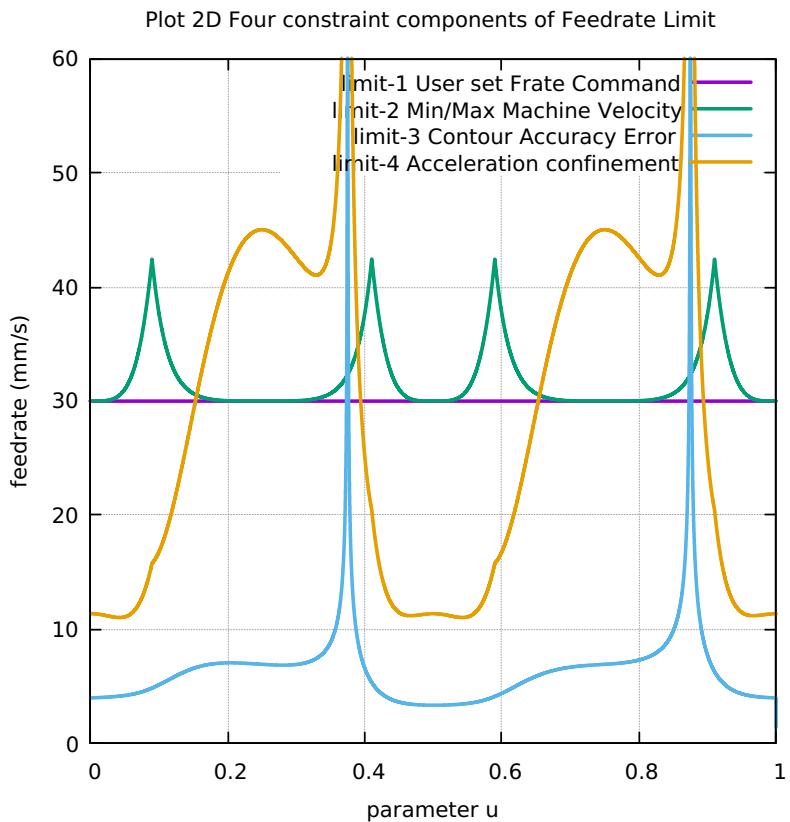


Figure 223: SkwAstroid FC40 Four Components FeedrateLimit

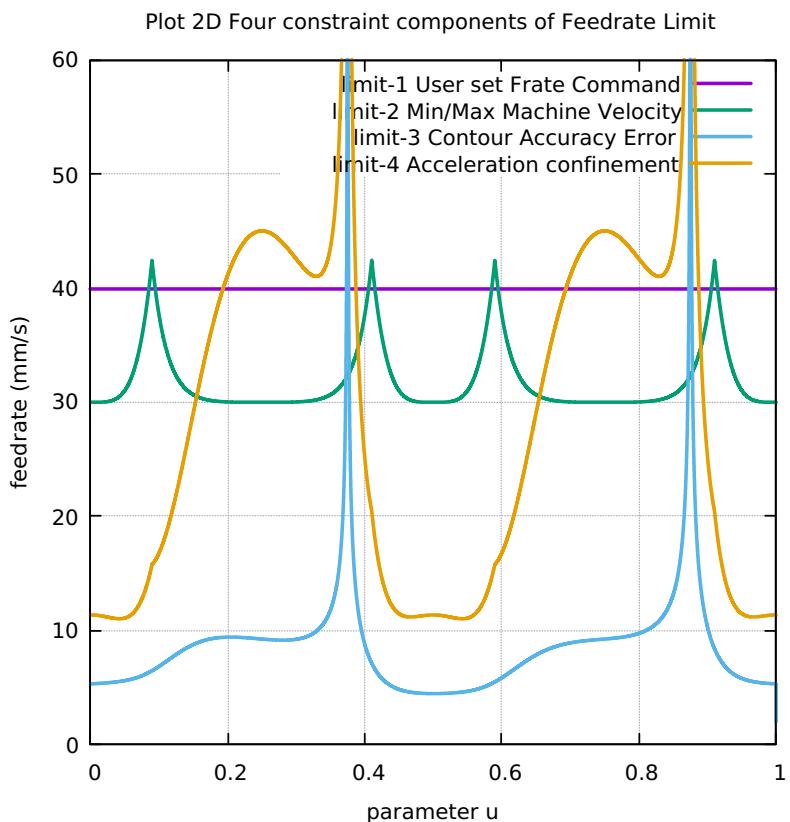


Figure 224: SkwAstroid Histogram Points FC10 FC20 FC30 FC40

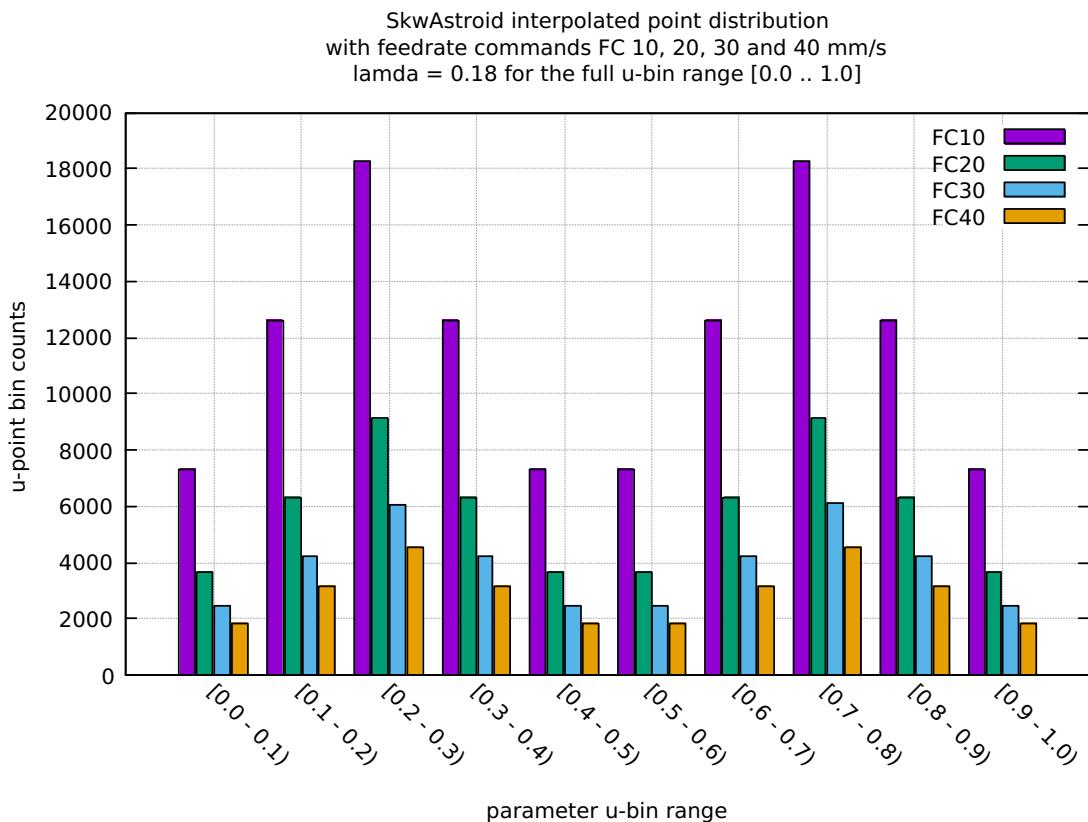


Table 12: SkwAstroid Table distribution of interpolated points

BINS	FC10	FC20	FC30	FC40
0.0 - 0.1	7322	3661	2441	1831
0.1 - 0.2	12603	6302	4202	3151
0.2 - 0.3	18245	9123	6081	4562
0.3 - 0.4	12604	6302	4202	3151
0.4 - 0.5	7322	3662	2442	1832
0.5 - 0.6	7323	3662	2442	1832
0.6 - 0.7	12603	6302	4201	3151
0.7 - 0.8	18244	9122	6082	4561
0.8 - 0.9	12604	6303	4202	3152
0.9 - 1.0	7324	3663	2443	1833
Tot Counts	116194	58102	38738	29056

Table 13: SkwAstroid Table FC10-20-30-40 Run Performance data

1	Curve Type	SKEWED-ASTROID	SKEWED-ASTROID	SKEWED-ASTROID
2	User Feedrate Command FC(mm/s)	FC10	FC20	FC40
3	User Lamda Acceleration Safety Factor	0.18	0.18	0.18
4	Total Interpolated Points (TIP)	116194	58102	38738
5	Total Sum-Chord-Error (SCE) (mm)	5.1636830433374E-04	1.032591177038E-03	1.548668518105E-03
6	Ratio 1 = (SCE/TIP) = Chord-Error/Point	4.444056908225E-09	1.777234775714E-08	3.997905150386E-08
7	Total Sum-Arc-Length (SAL) (mm)	1.161845691077E+03	1.161851349542E+03	1.161862568638E+03
8	Total Sum-Chord-Length (SCL) (mm)	4.4571428558819E+02	4.457142855374E+02	4.457142846207E+02
9	Difference = (SAL - SCL) (mm)	7.161314051951E+02	7.161370640044E+02	7.161426907744E+02
10	Percentage Difference = (SAL - SCL)/SAL	6.163739390653E+01	6.163758077028E+01	6.163776660469E+01
11	Ratio 2 = (SCE/SCL) = Chord Error/Chord-Length	1.158518631091E-06	2.316710975042E-06	3.474576811966E-06
12	Total Sum-Arc-Theta (SAT) (rad)	8.421542473789E+00	8.421464824576E+00	8.421387277982E+00
13	Total Sum-Arc-Area (SAA) (mm2)	4.343191363991E-05	1.736583480748E-04	3.905753943512E-04
14	Ratio 3 = (SAA/SCL) = Arc-Area/Chord-Length	1.158518631091E-06	2.316710975042E-06	3.474576811966E-06
15	Average-Chord-Error (ACE) (mm)	4.444056908225E-09	1.777234775714E-08	3.997905150386E-08
16	Average-Arc-Length (AAL) (mm)	9.999274406178E-03	1.999709728820E-02	2.999346814144E-02
17	Average-Chord-Length (ACL) (mm)	3.835982252648E-03	7.671370295474E-03	1.150616425177E-02
18	Average-Arc-Theta (AAT) (rad)	7.247891416685E-05	1.449452647042E-04	2.173990571800E-04
19	Average-Arc-Area (AAA) (mm2)	3.737911375032E-10	2.988904632878E-09	1.008274761471E-08
20	Algorithm actual runtime on computer (ART) (s)	38.427006969	19.305250287	12.910301226
				9.663656563

.9 APPENDIX RIBBON-10L CURVE

- .9.1 Plot of Ribbon-10L curve [225]**
- .9.2 Ribbon-10L Radius of Curvature [226]**
- .9.3 Ribbon-10L Validation in LinuxCNC [227]**
- .9.4 Ribbon-10L Direction of Travel 3D [228]**
- .9.5 Ribbon-10L First and Second Order Taylors App [229]**
- .9.6 Ribbon-10L First minus Second Order Taylors App [230]**
- .9.7 Ribbon-10L Separate First Second Order Taylors App [??]**
- .9.8 Ribbon-10L Separation SAL and SCL [232]**
- .9.9 Ribbon-10L Chord-error in close view 2 scales [233]**
- .9.10 Ribbon-10L Four Components Feedrate Limit [234]**
- .9.11 Ribbon-10L FrateCommand FrateLimit and Curr-Frate [235]**
- .9.12 Ribbon-10L FeedRateLimit minus CurrFeedRate [236]**
- .9.13 Ribbon-10L FC20-Nominal X and Y Feedrate Profiles [237]**
- .9.14 Ribbon-10L FC20 Nominal Tangential Acceleration [238]**
- .9.15 Ribbon-10L FC20 Nominal Rising S-Curve Profile [239]**
- .9.16 Ribbon-10L FC20 Nominal Falling S-Curve Profile [240]**
- .9.17 Ribbon-10L FC10 Colored Feedrate Profile ngcode [241]**
- .9.18 Ribbon-10L FC20 Colored Feedrate Profile ngcode [242]**

- .9.19 **Ribbon-10L FC30 Colored Feedrate Profile ngcode [243]**
- .9.20 **Ribbon-10L FC40 Colored Feedrate Profile ngcode [244]**
- .9.21 **Ribbon-10L FC10 Tangential Acceleration [245]**
- .9.22 **Ribbon-10L FC20 Tangential Acceleration [246]**
- .9.23 **Ribbon-10L FC30 Tangential Acceleration [247]**
- .9.24 **Ribbon-10L FC40 Tangential Acceleration [248]**
- .9.25 **Ribbon-10L FC20 Nominal Separation NAL and NCL [249]**
- .9.26 **Ribbon-10L SAL minus SCL for FC10 FC20 FC30 FC40 [??]**
- .9.27 **Ribbon-10L FC10 FrateCmd CurrFrate X-Frate Y-Frate [251]**
- .9.28 **Ribbon-10L FC20 FrateCmd CurrFrate X-Frate Y-Frate [252]**
- .9.29 **Ribbon-10L FC30 FrateCmd CurrFrate X-Frate Y-Frate [253]**
- .9.30 **Ribbon-10L FC40 FrateCmd CurrFrate X-Frate Y-Frate [254]**
- .9.31 **Ribbon-10L FC10 Four Components FeedrateLimit [255]**
- .9.32 **Ribbon-10L FC20 Four Components FeedrateLimit [256]**
- .9.33 **Ribbon-10L FC30 Four Components FeedrateLimit [257]**
- .9.34 **Ribbon-10L FC40 Four Components FeedrateLimit [258]**
- .9.35 **Ribbon-10L Histogram Points FC10 FC20 FC30 FC40 [259]**
- .9.36 **Ribbon-10L Table distribution of interpolated points [14]**
- .9.37 **Ribbon-10L Table FC10-20-30-40 Run Performance data [15]**

Figure 225: Plot of Ribbon-10L curve

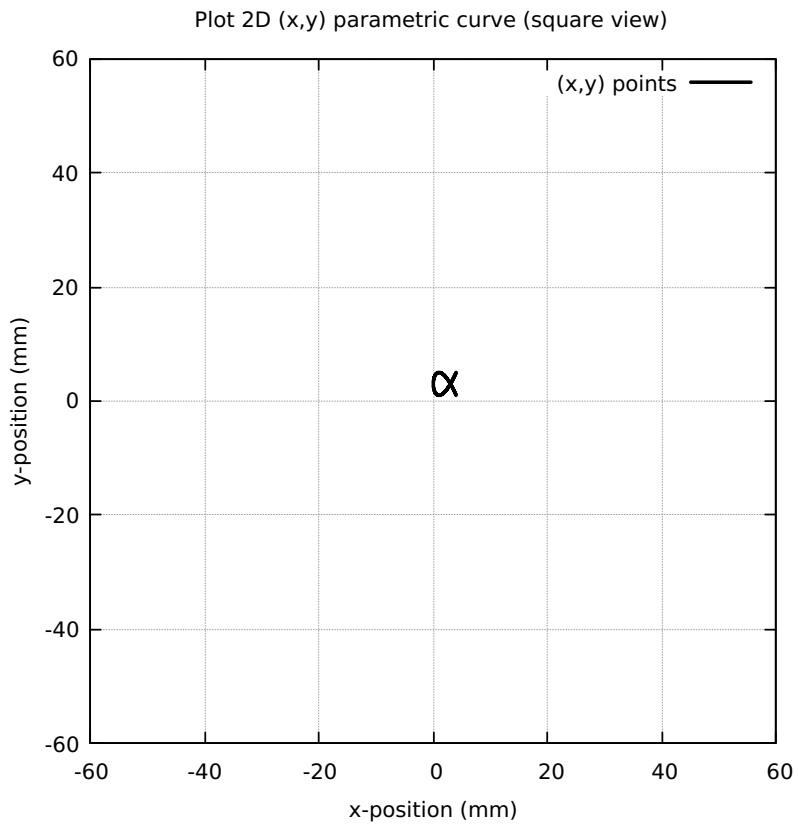


Figure 226: Ribbon-10L Radius of Curvature

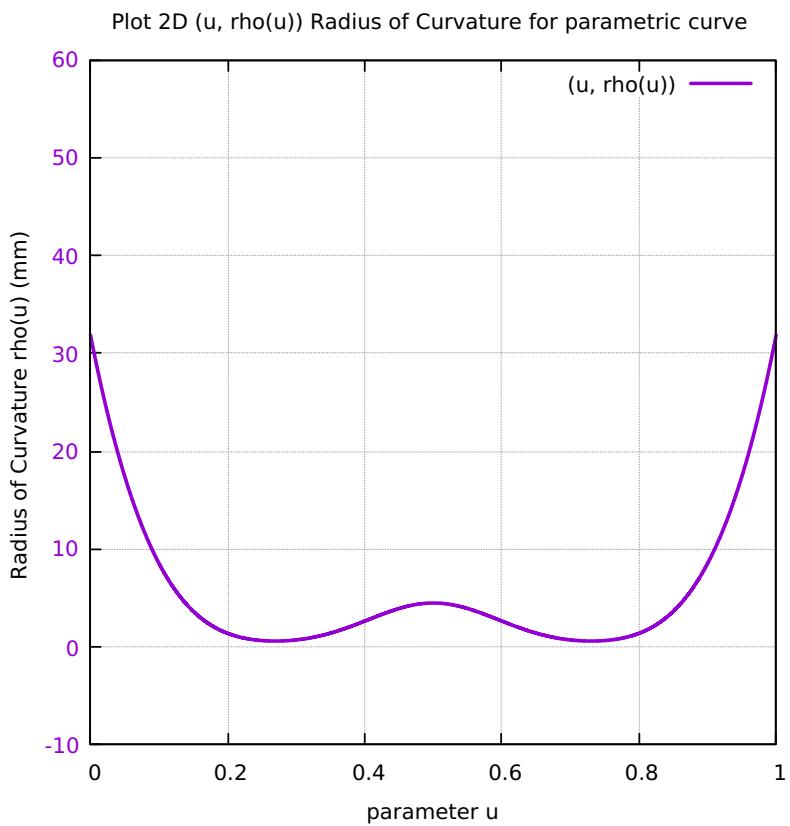


Figure 227: Ribbon-10L Validation in LinuxCNC

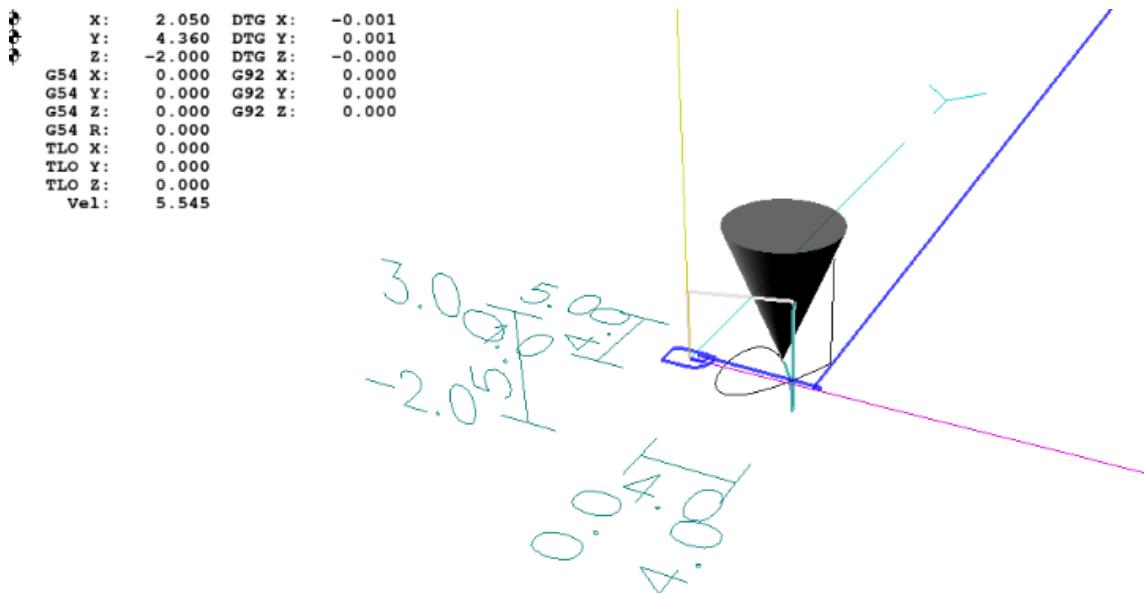


Figure 228: Ribbon-10L Direction of Travel 3D

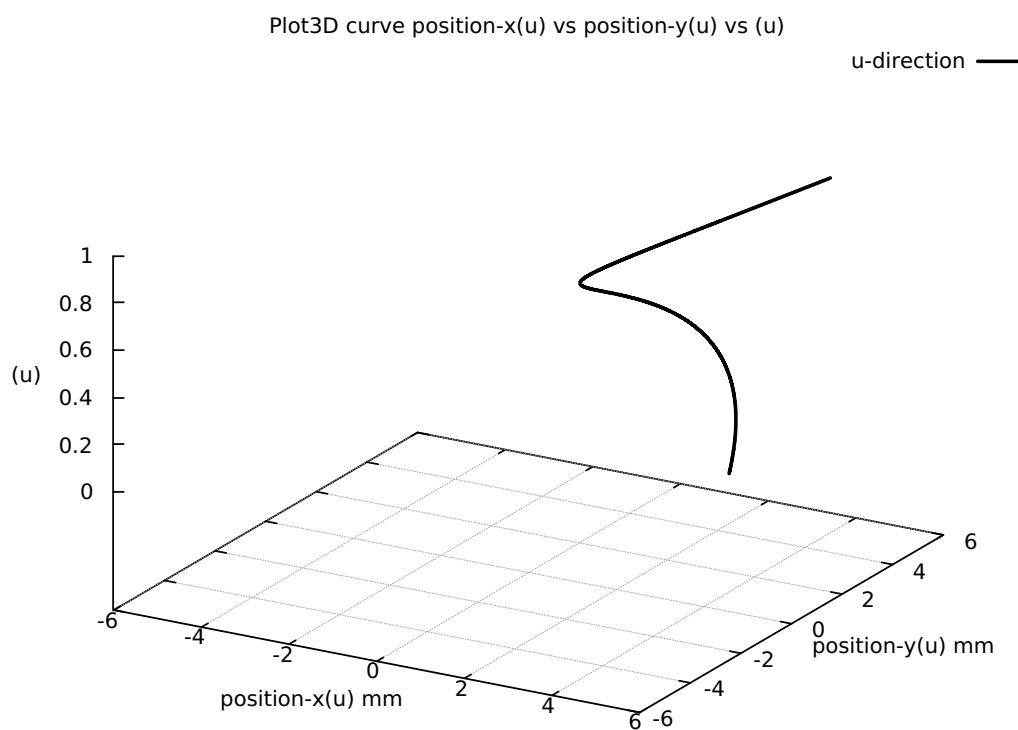


Figure 229: Ribbon-10L First and Second Order Taylor's Approximation

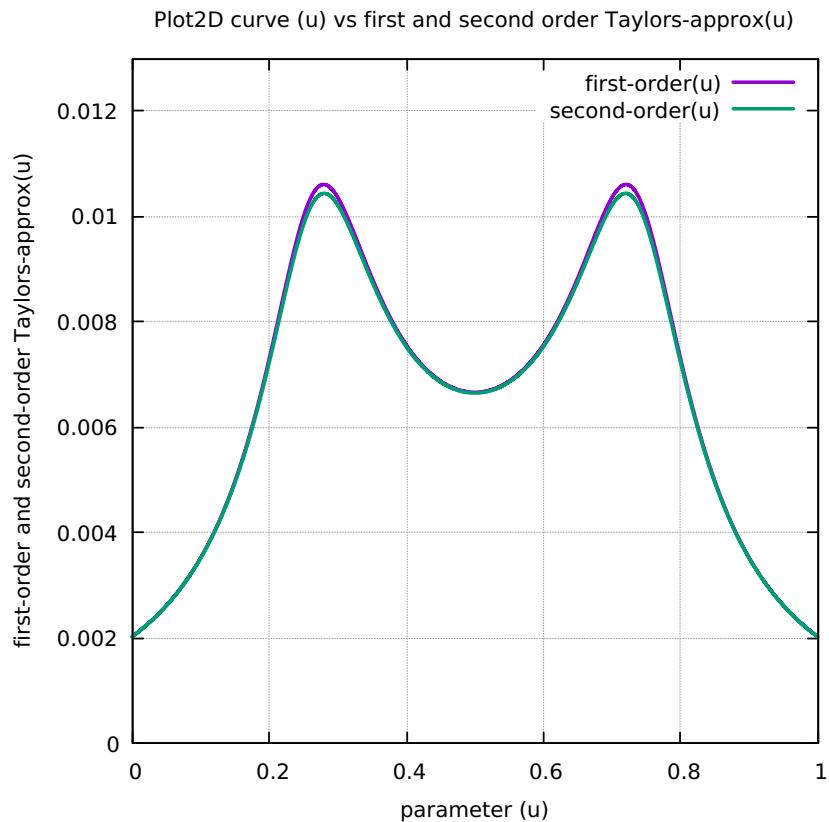


Figure 230: Ribbon-10L First minus Second Order Taylor's Approximation

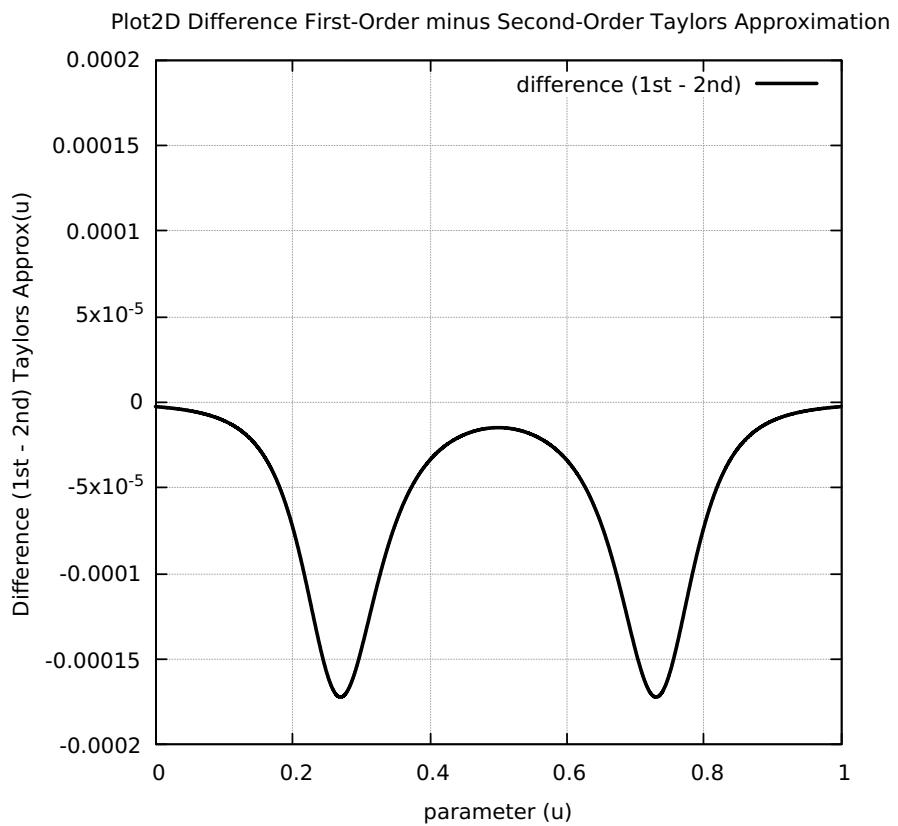


Figure 231: Ribbon-10L First and Second Order Taylor's Approximation

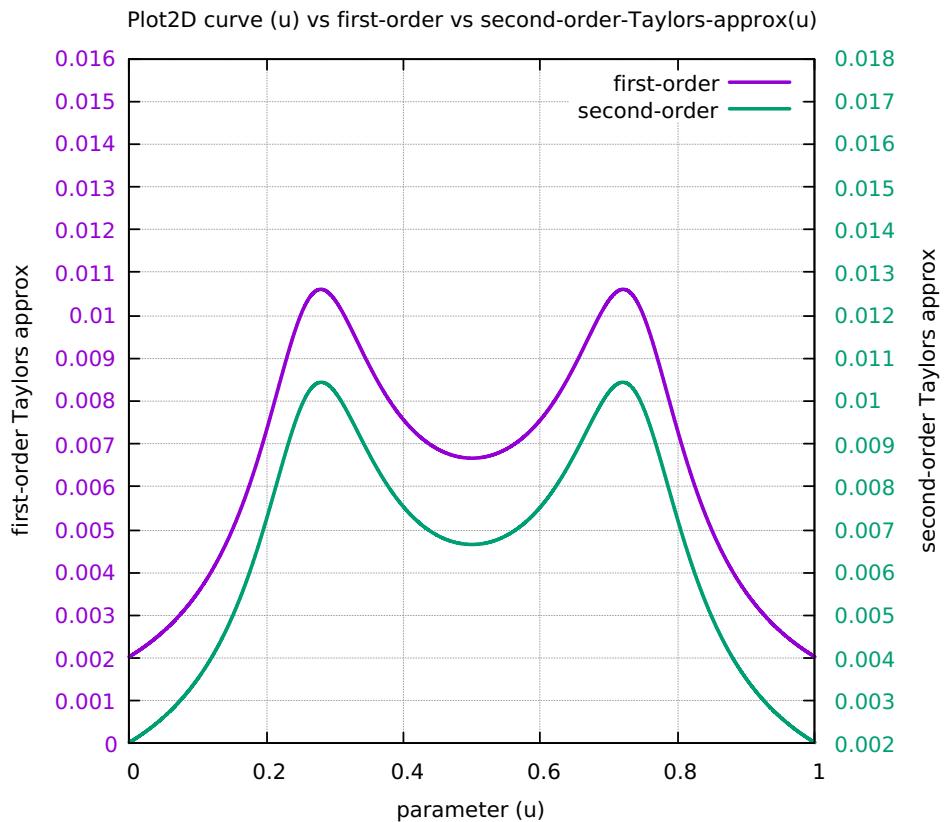


Figure 232: Ribbon-10L Separation SAL and SCL

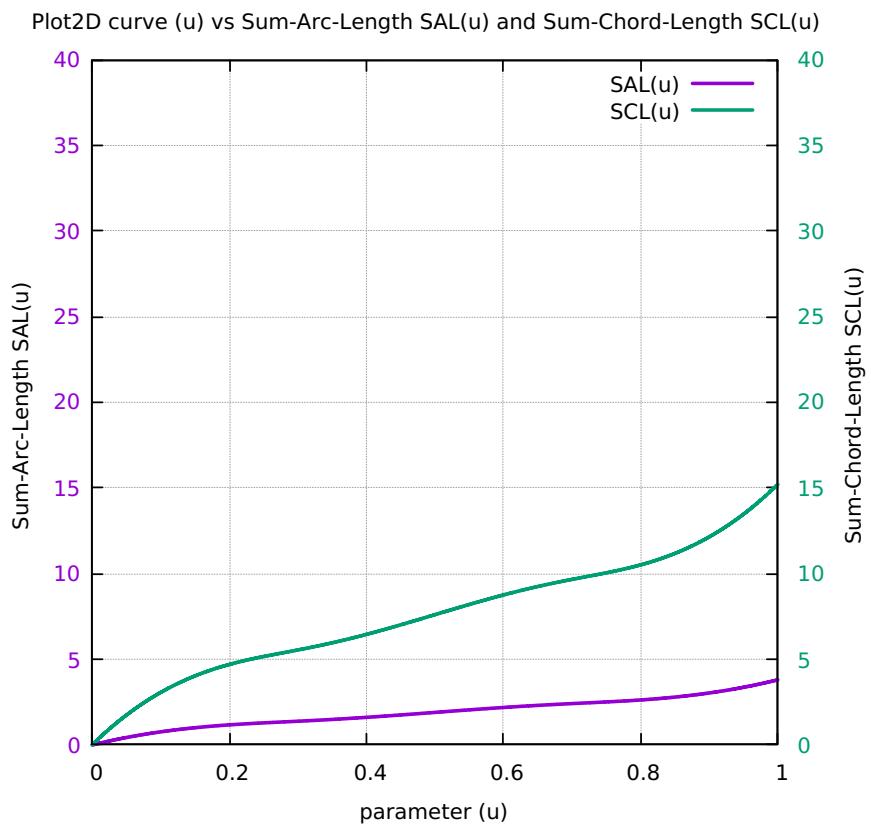


Figure 233: Ribbon-10L Chord-error in close view 2 scales

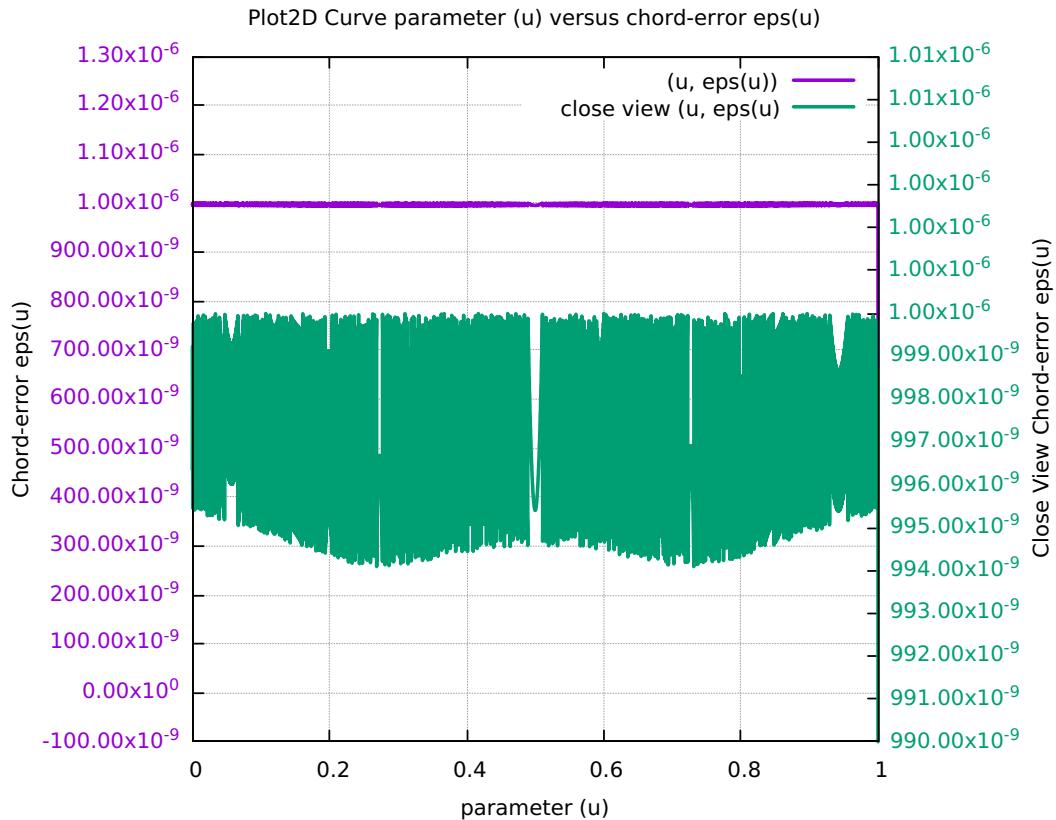


Figure 234: Ribbon-10L Four Components Feedrate Limit

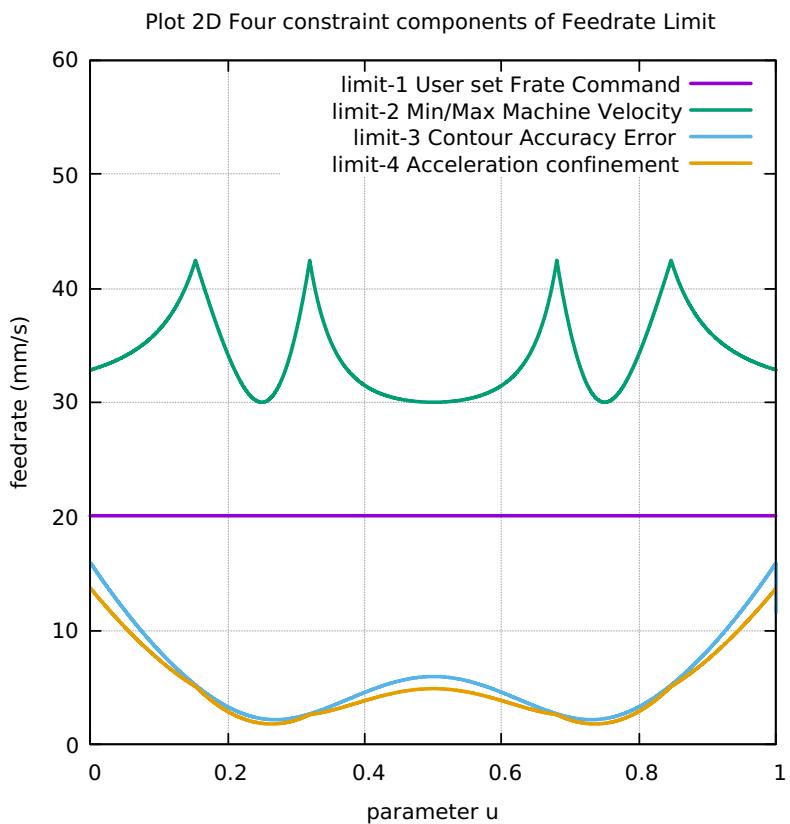


Figure 235: Ribbon-10L FrateCommand FrateLimit and Curr-Frate

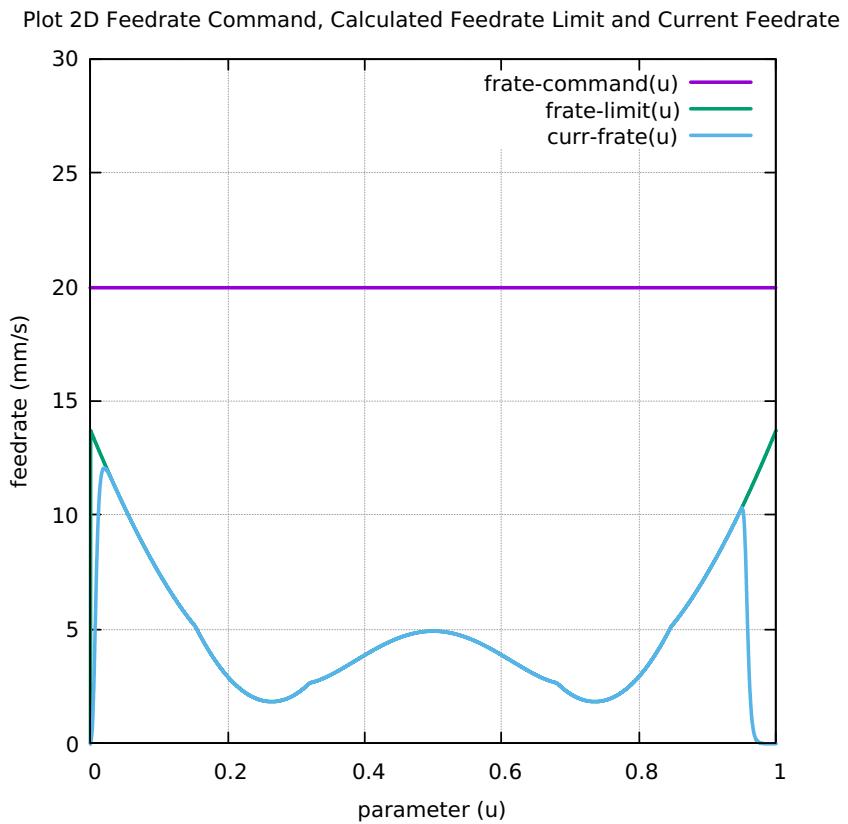


Figure 236: Ribbon-10L FeedRateLimit minus CurrFeedRate

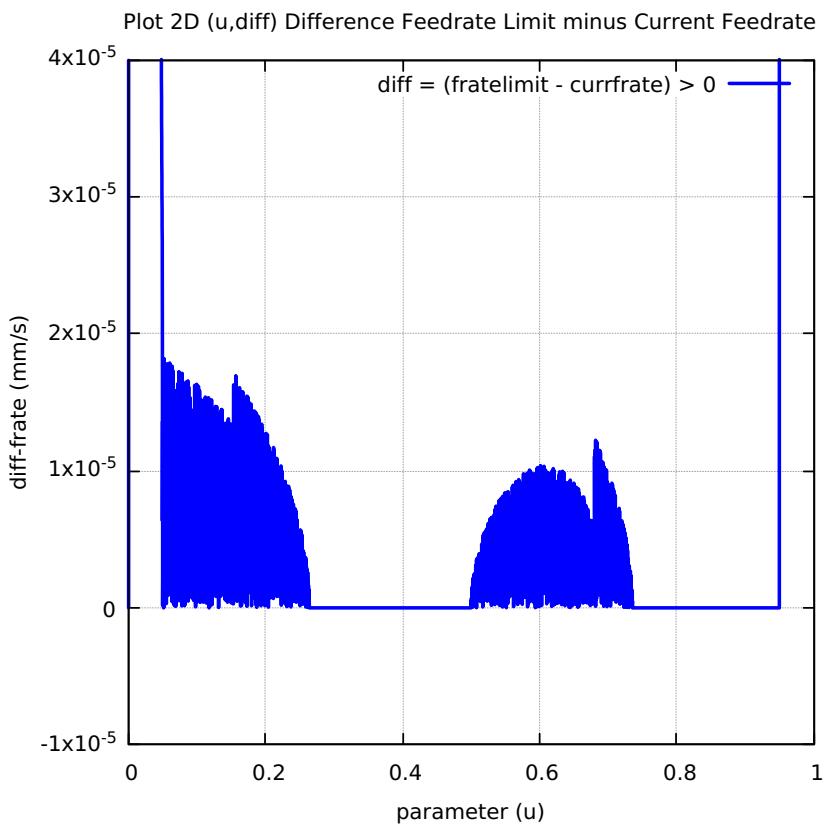


Figure 237: Ribbon-10L FC20-Nominal X and Y Feedrate Profiles

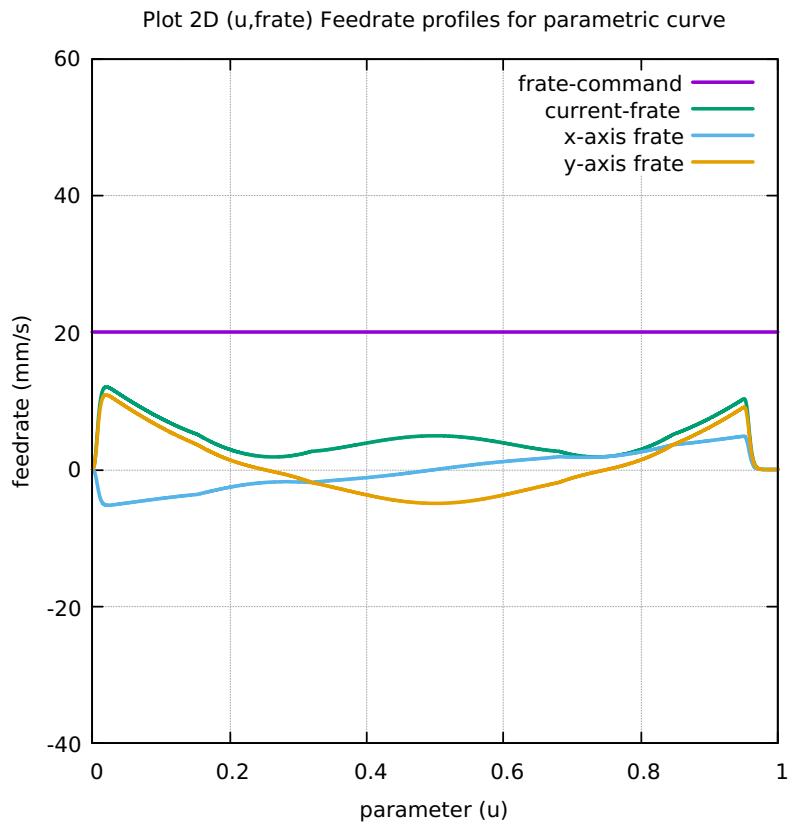


Figure 238: Ribbon-10L FC20 Nominal Tangential Acceleration

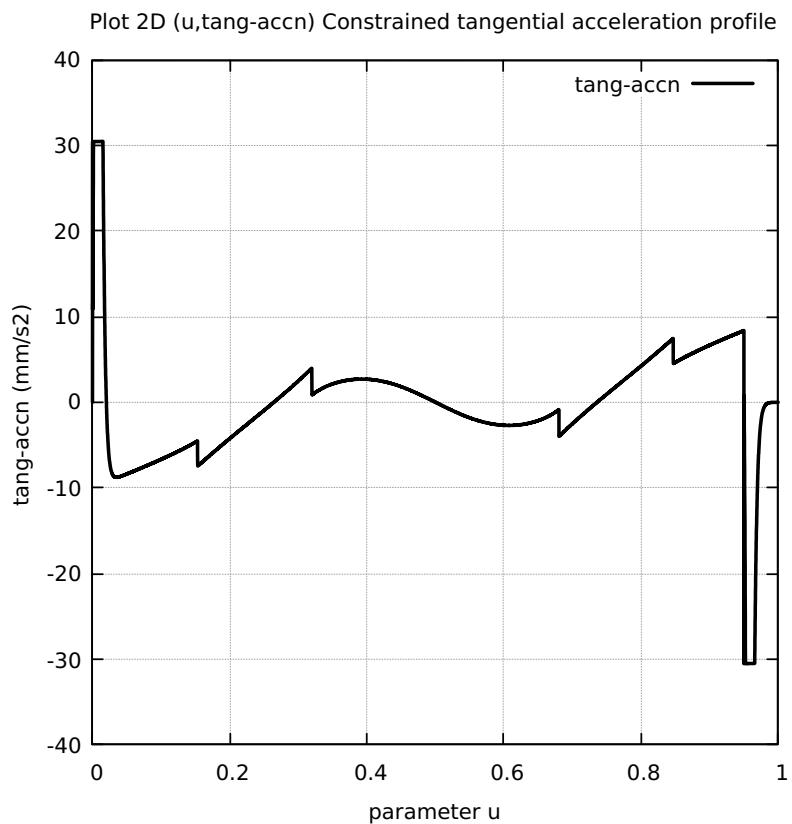


Figure 239: Ribbon-10L FC20 Nominal Rising S-Curve Profile

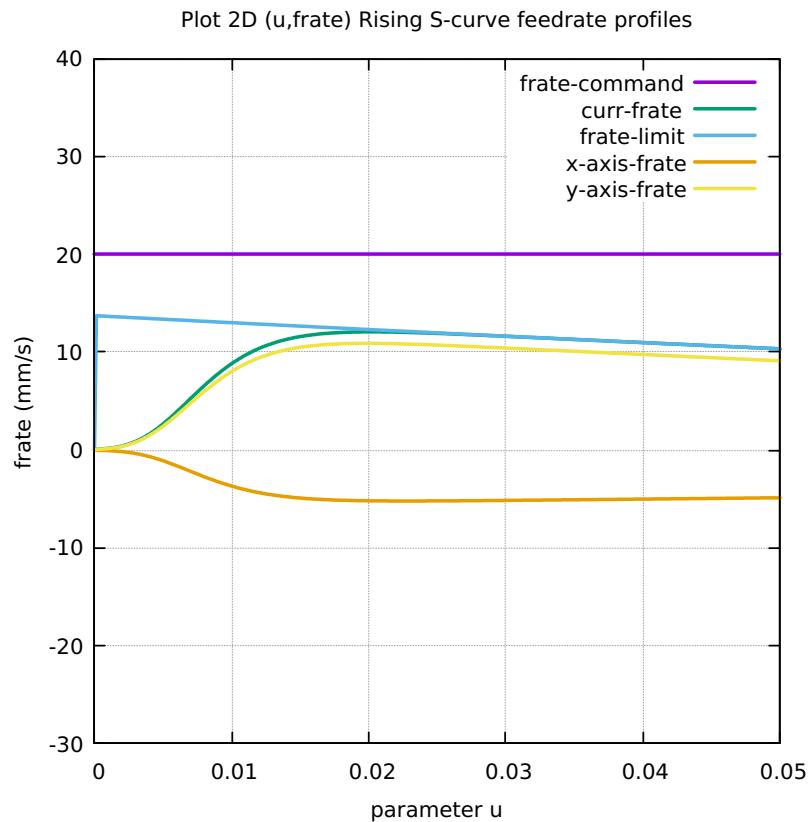


Figure 240: Ribbon-10L FC20 Nominal Falling S-Curve Profile

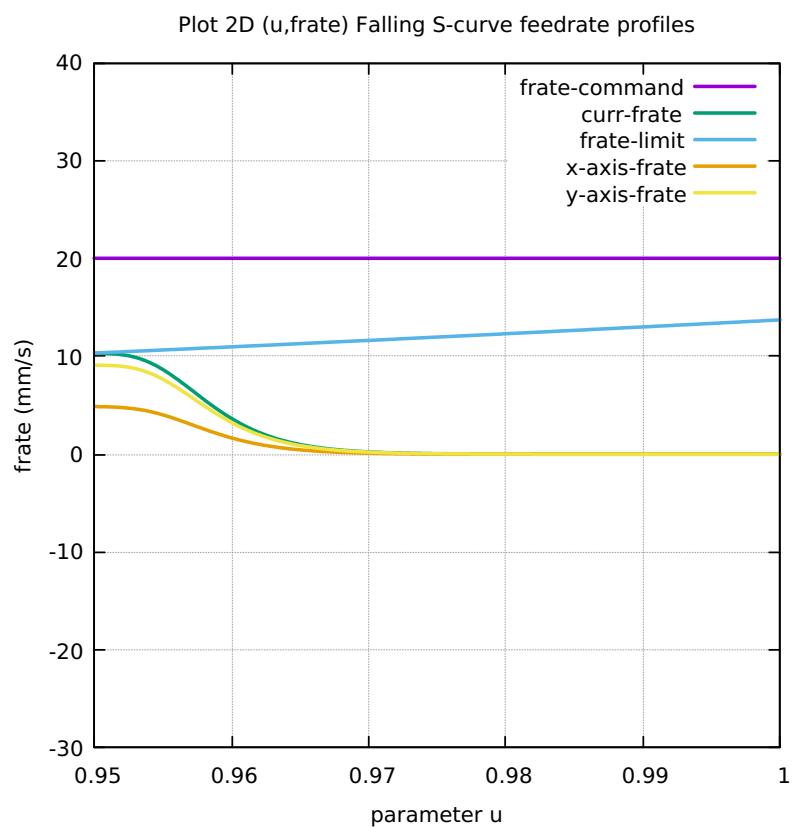


Figure 241: Ribbon-10L FC10 Colored Feedrate Profile data ngcode

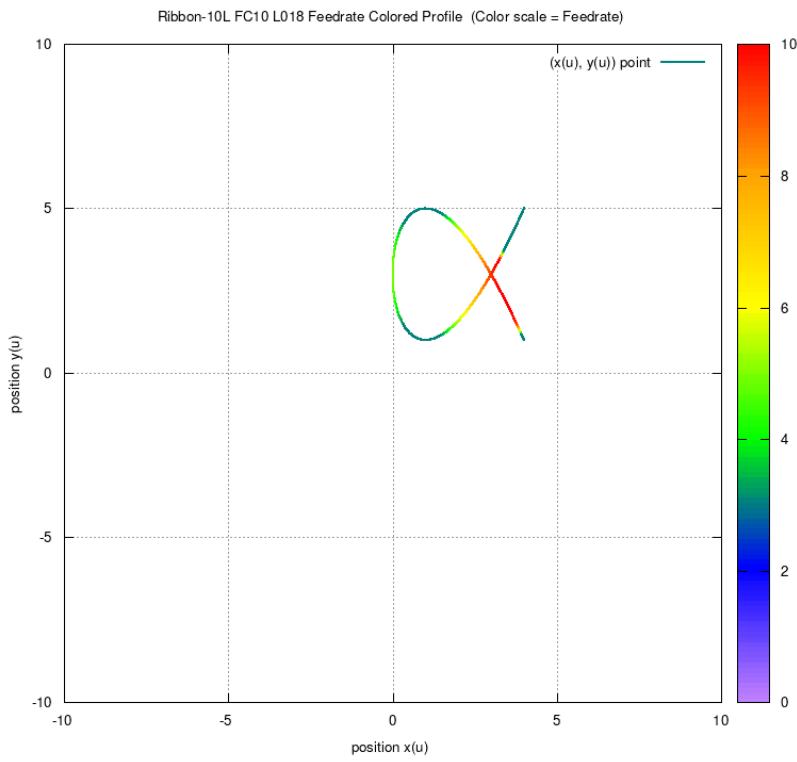


Figure 242: Ribbon-10L FC20 Colored Feedrate Profile data ngcode

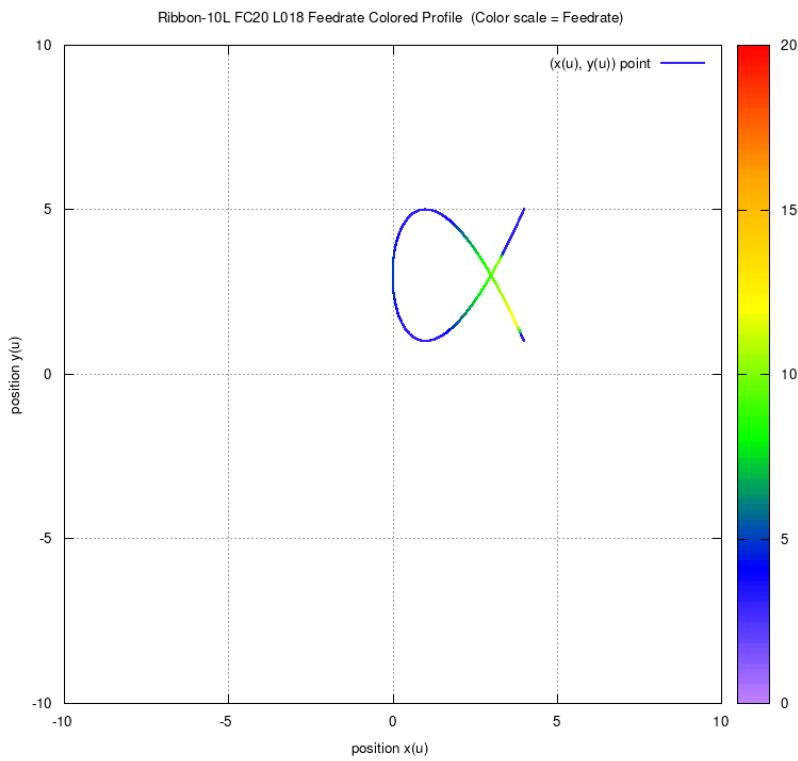


Figure 243: Ribbon-10L FC30 Colored Feedrate Profile data ngcode

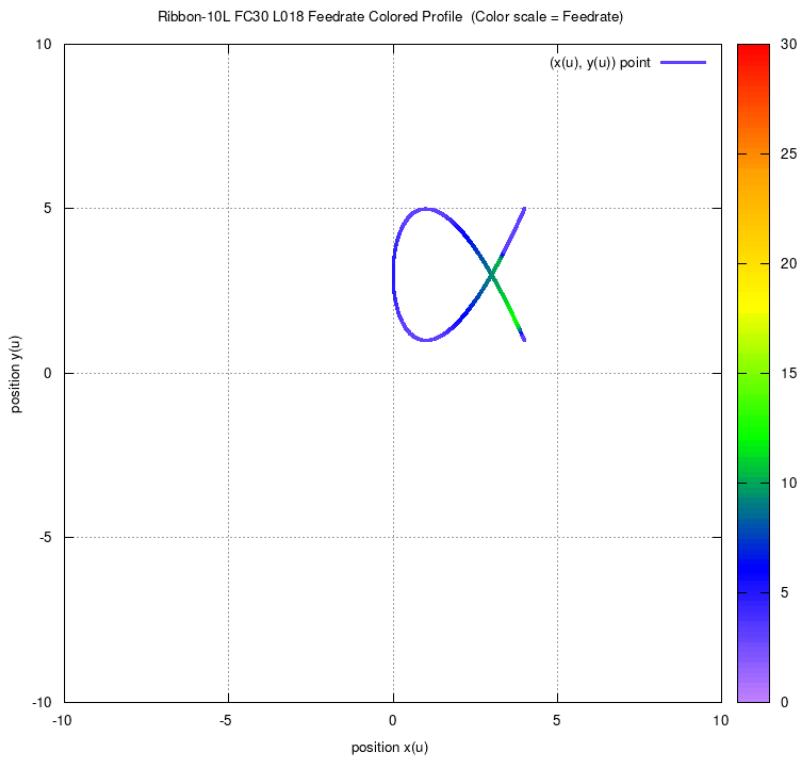


Figure 244: Ribbon-10L FC40 Colored Feedrate Profile data ngcode

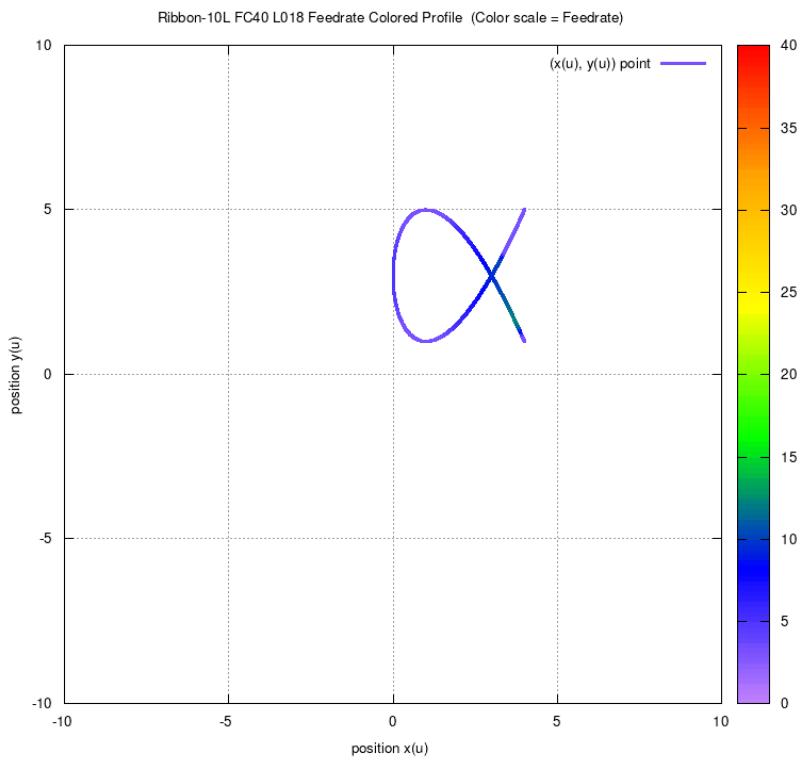


Figure 245: Ribbon-10L FC10 Tangential Acceleration

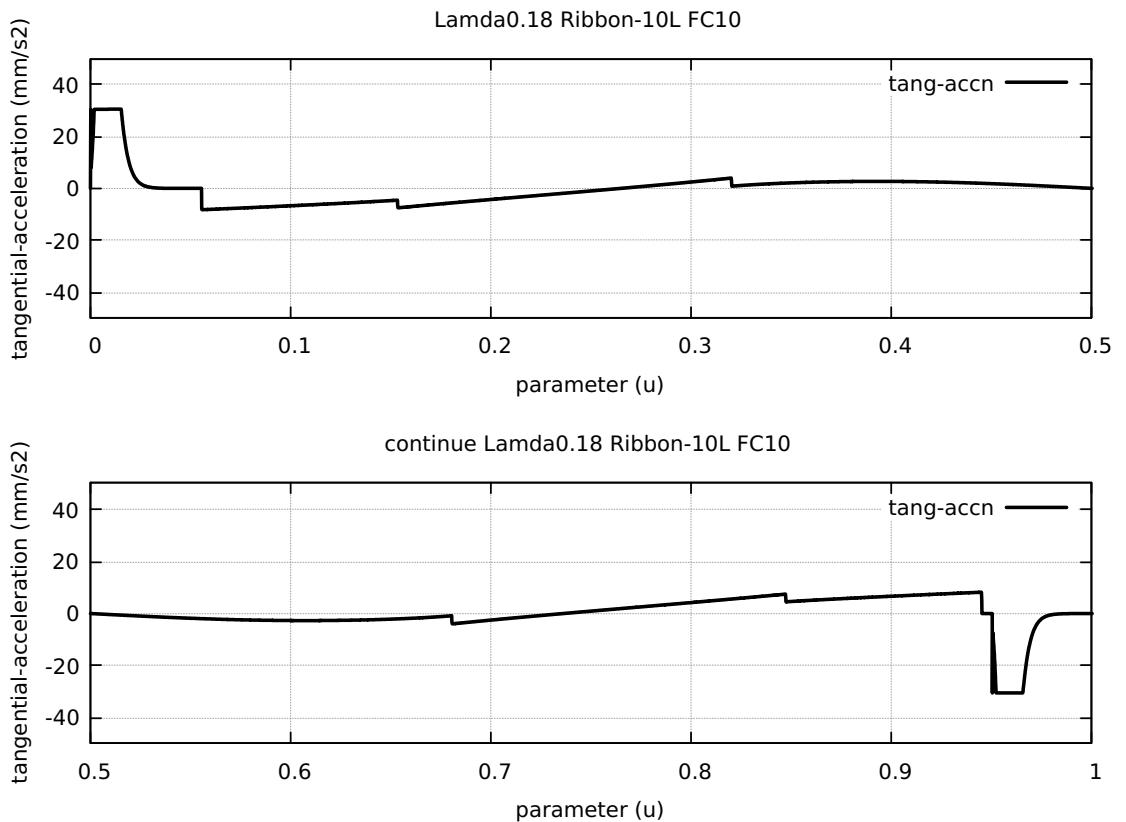


Figure 246: Ribbon-10L FC20 Tangential Acceleration

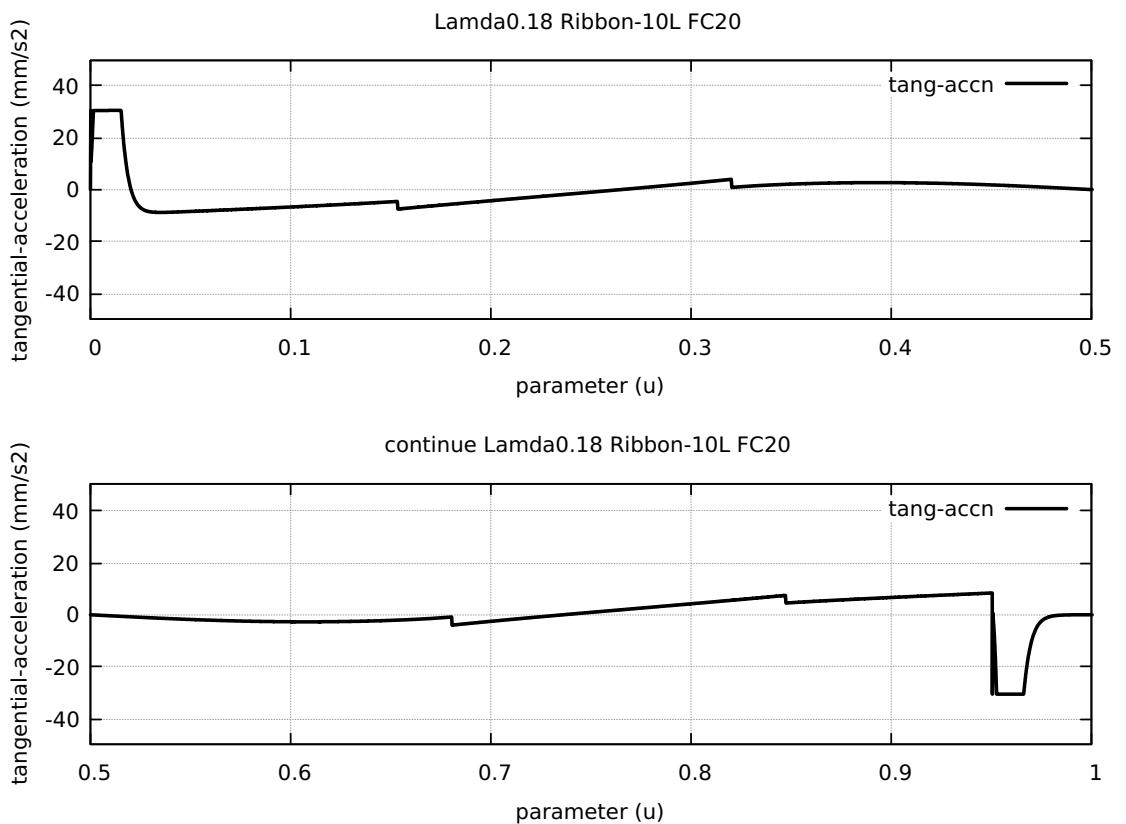


Figure 247: Ribbon-10L FC30 Tangential Acceleration

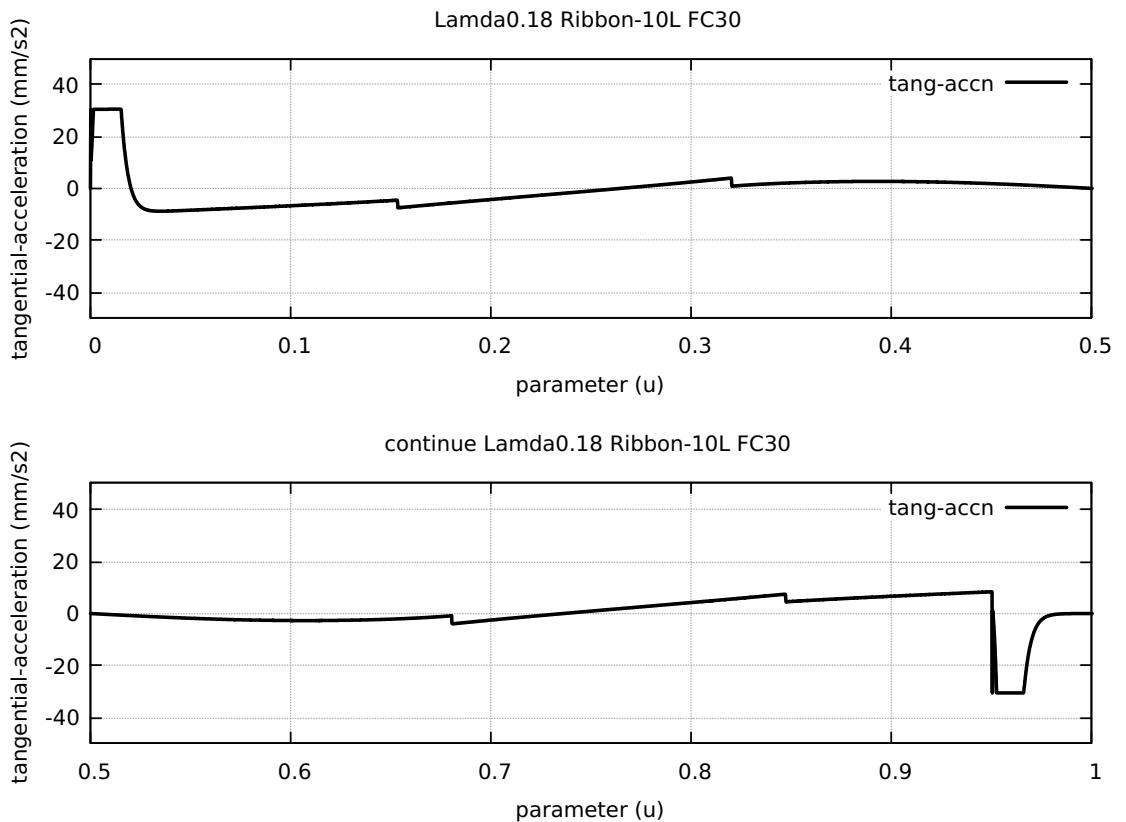


Figure 248: Ribbon-10L FC40 Tangential Acceleration

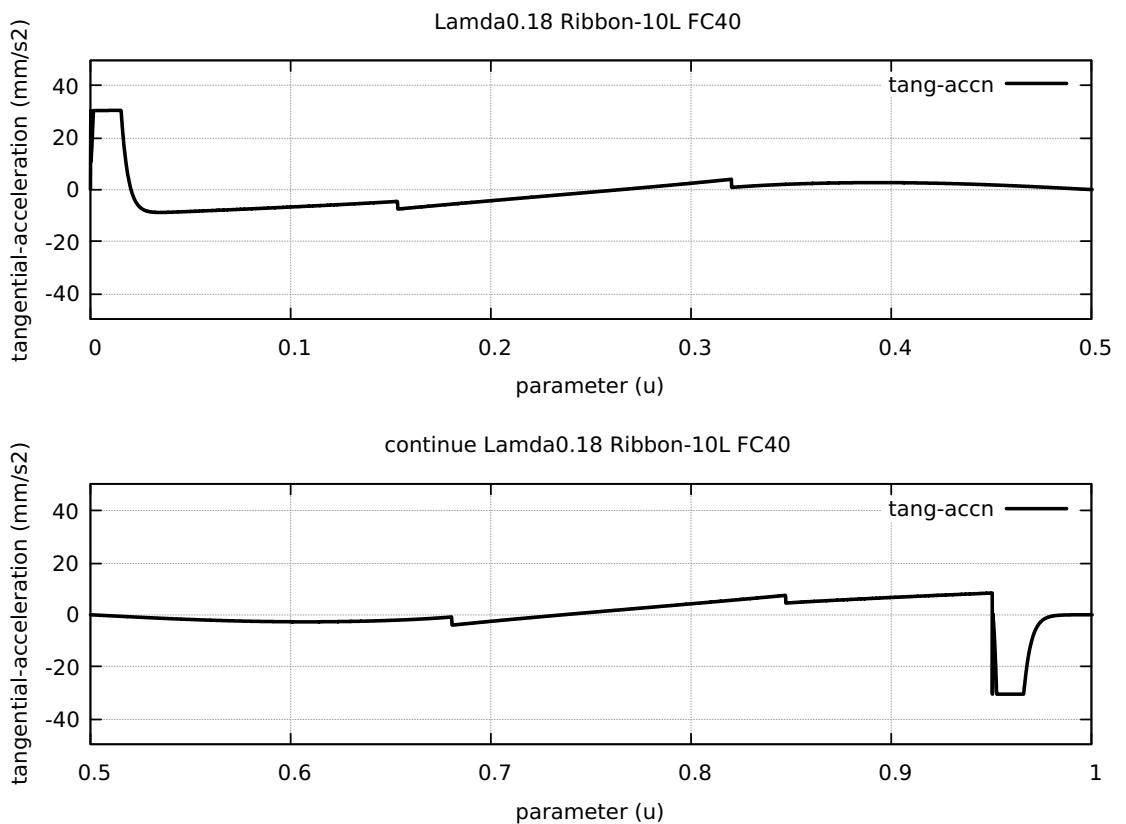


Figure 249: Ribbon-10L FC20 Nominal Separation NAL and NCL

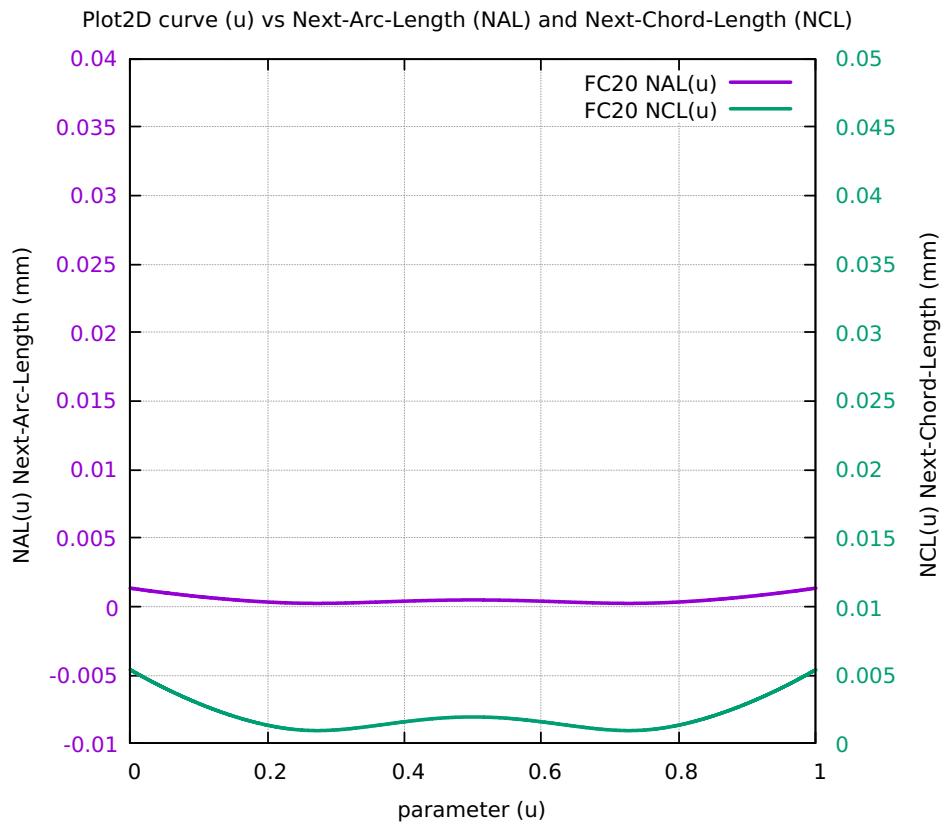


Figure 250: Ribbon-10L SAL minus SCL for FC10 FC20 FC30 FC40

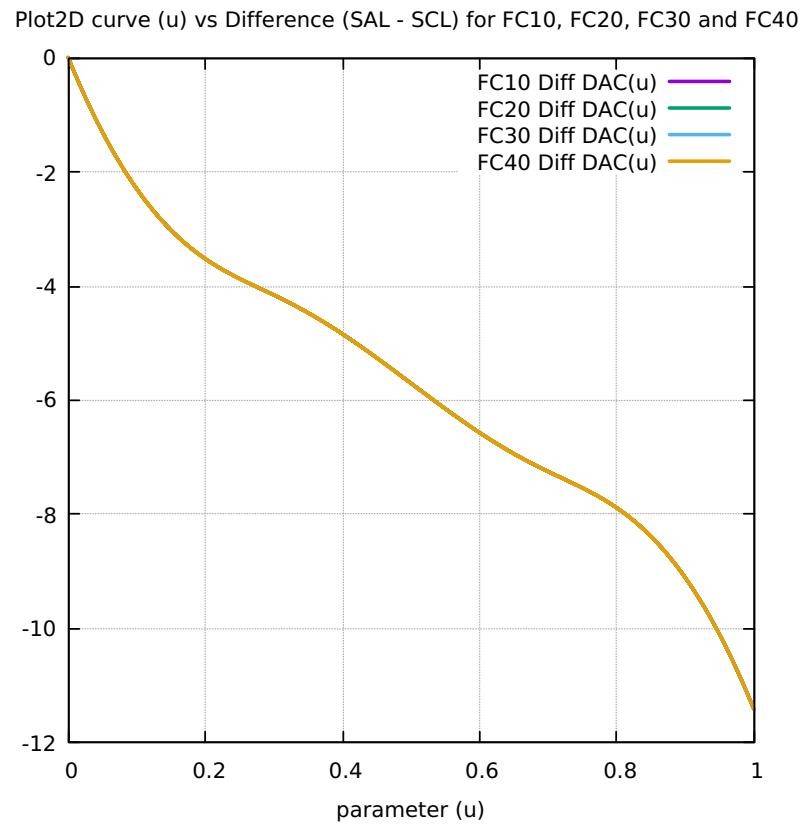


Figure 251: Ribbon-10L FC10 FrateCmd CurrFrate X-Frate Y-Frate

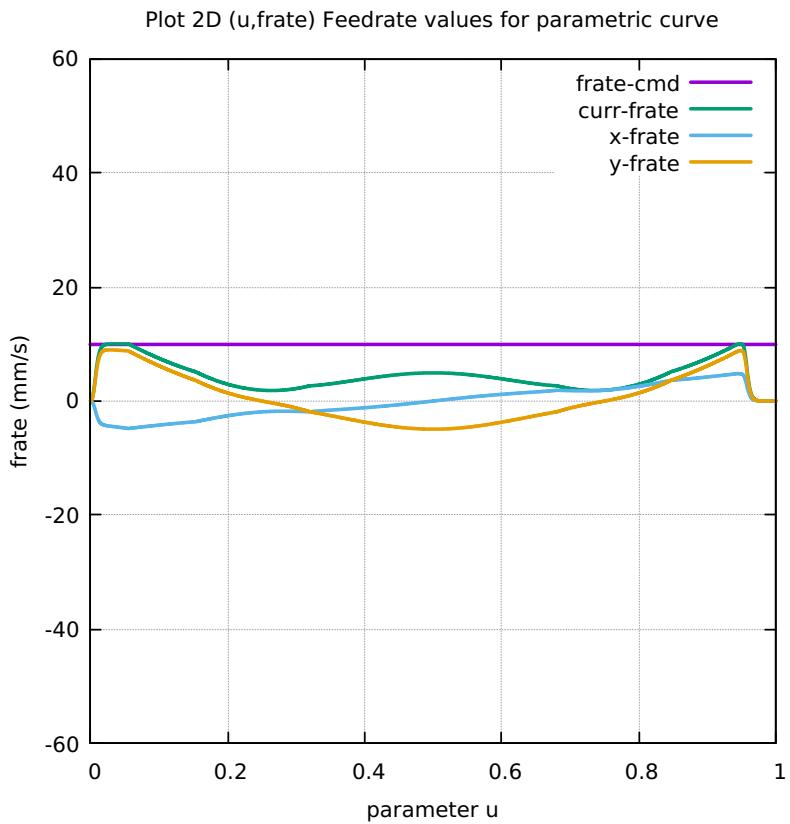


Figure 252: Ribbon-10L FC20 FrateCmd CurrFrate X-Frate Y-Frate

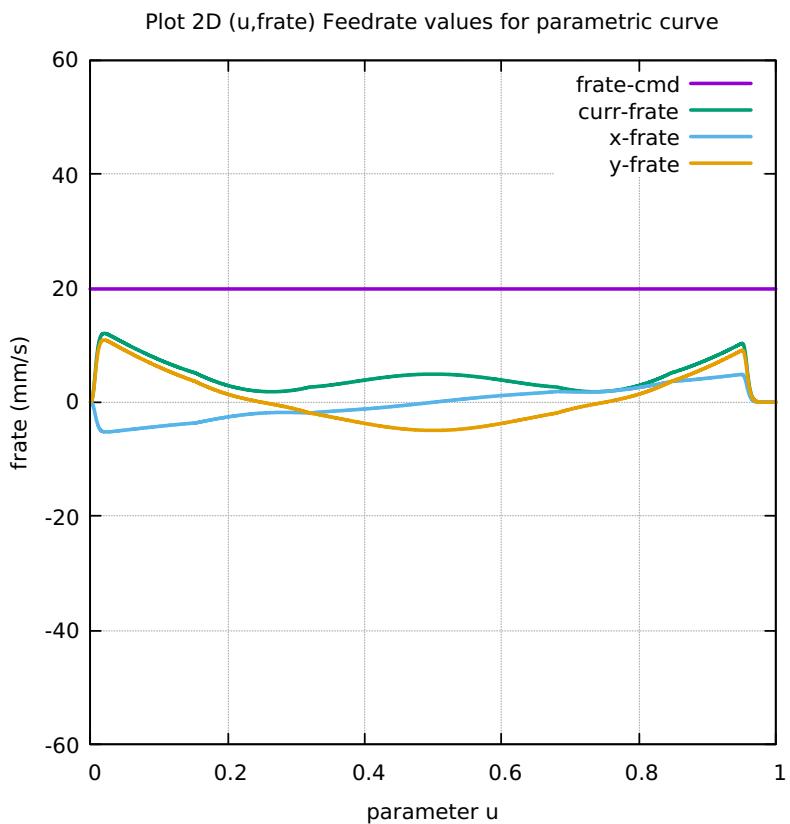


Figure 253: Ribbon-10L FC30 FrateCmd CurrFrate X-Frate Y-Frate

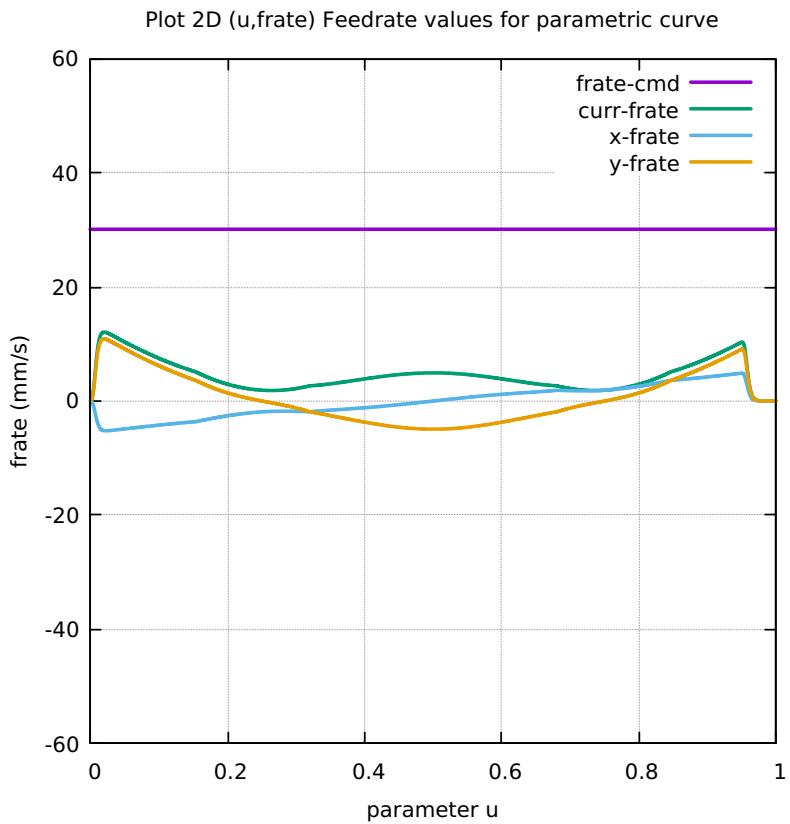


Figure 254: Ribbon-10L FC40 FrateCmd CurrFrate X-Frate Y-Frate

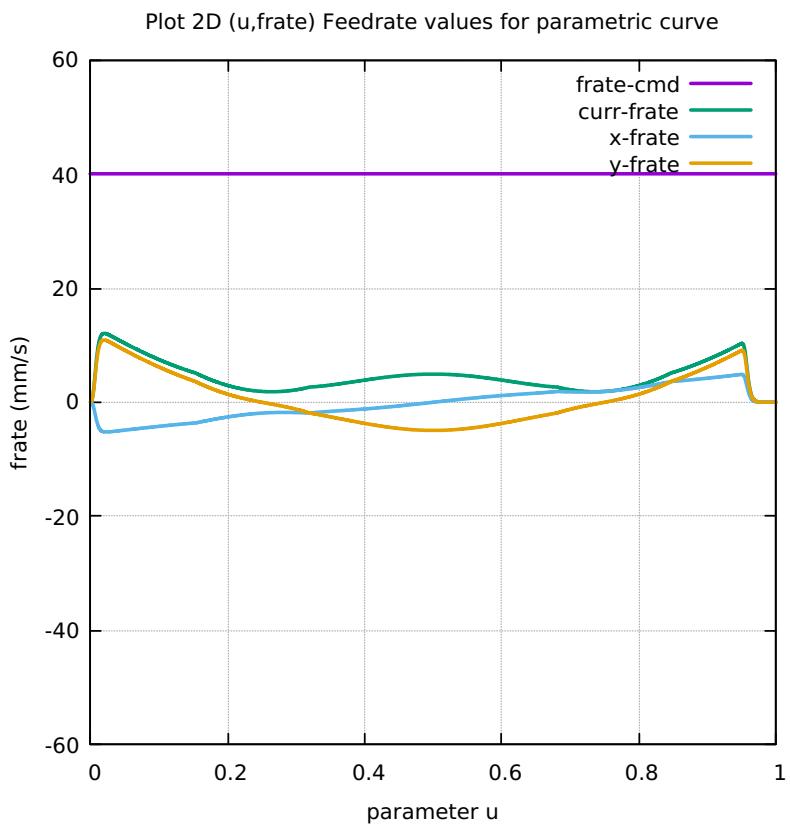


Figure 255: Ribbon-10L FC10 Four Components FeedrateLimit

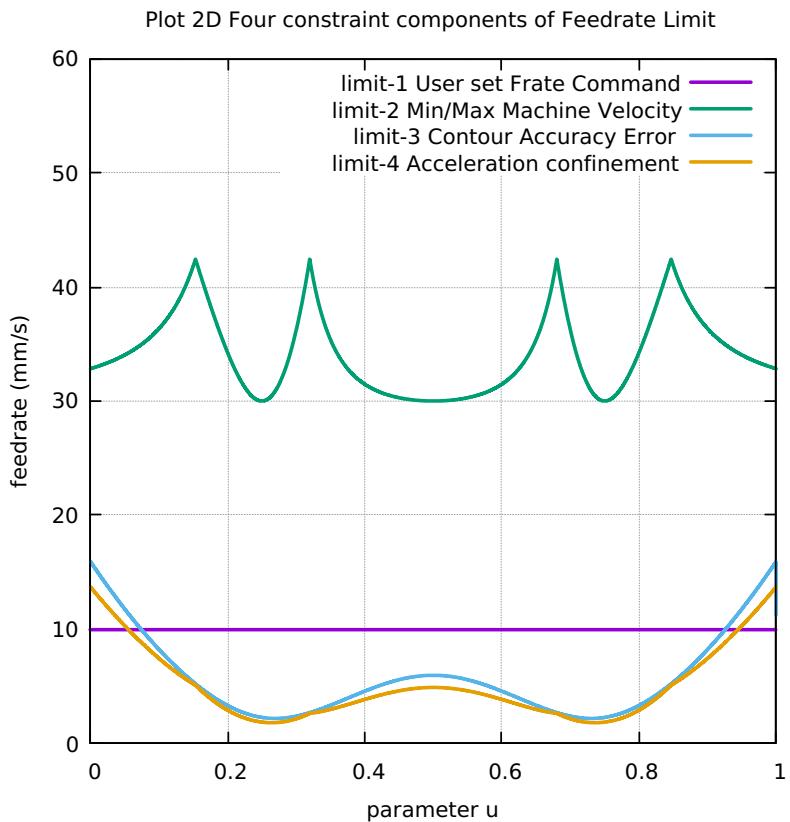


Figure 256: Ribbon-10L FC20 Four Components FeedrateLimit

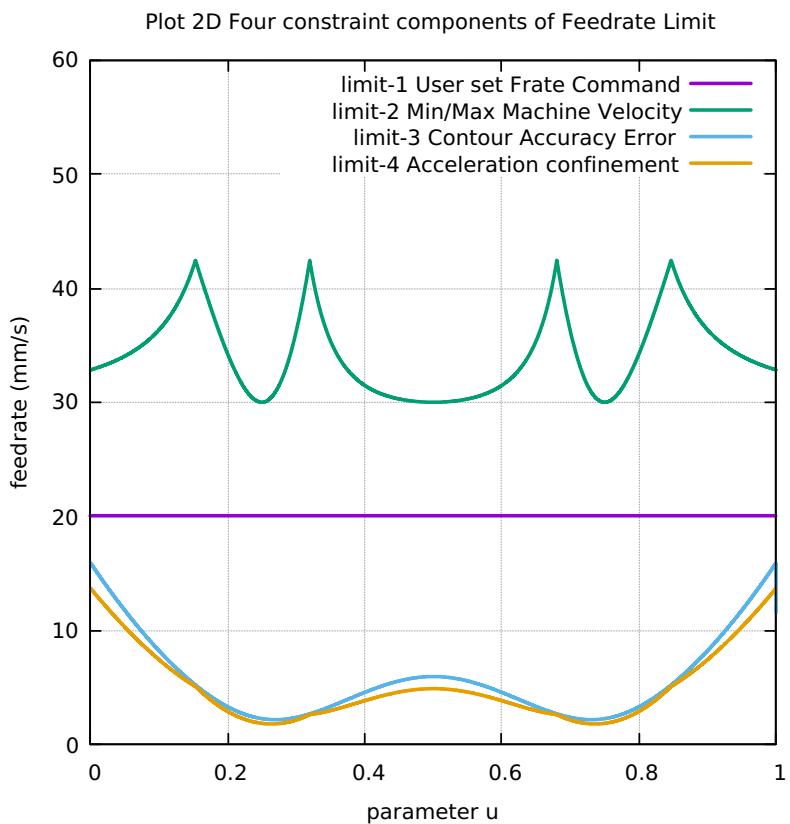


Figure 257: Ribbon-10L FC30 Four Components FeedrateLimit

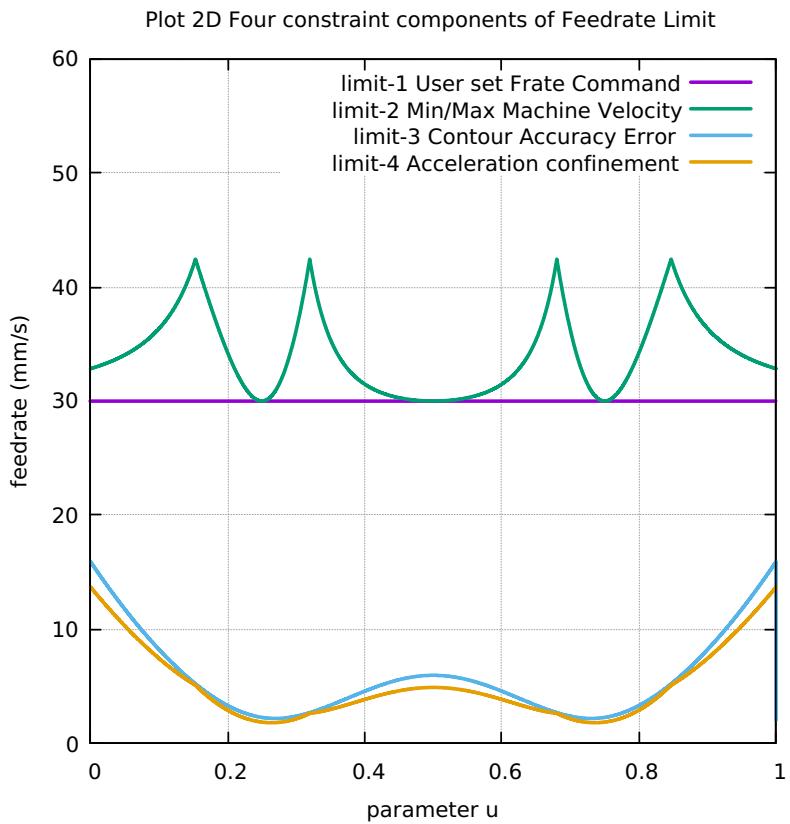


Figure 258: Ribbon-10L FC40 Four Components FeedrateLimit

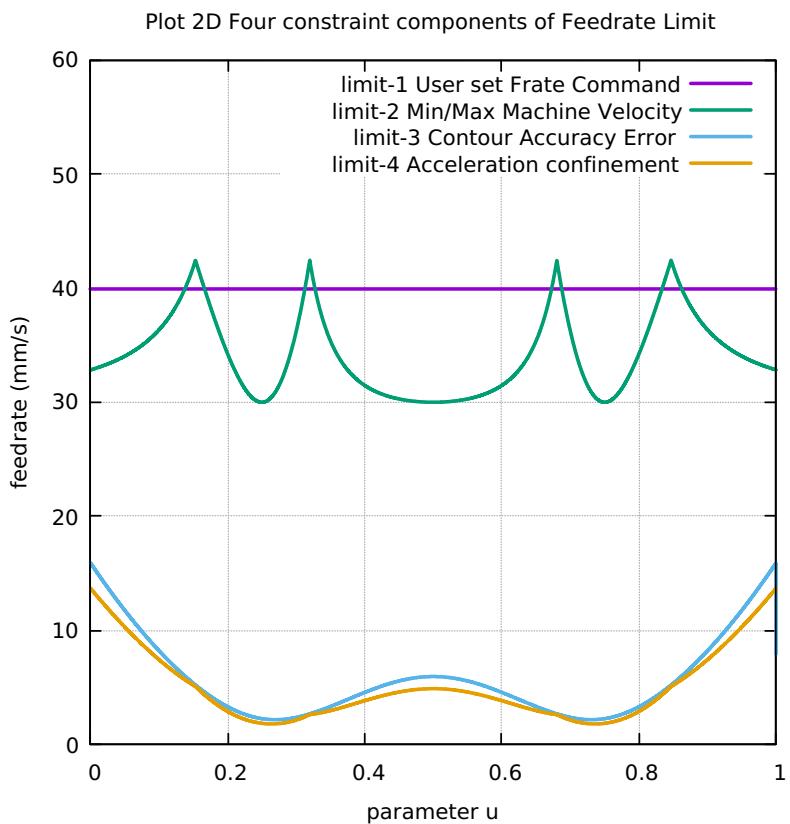


Figure 259: Ribbon-10L Histogram Points FC10 FC20 FC30 FC40

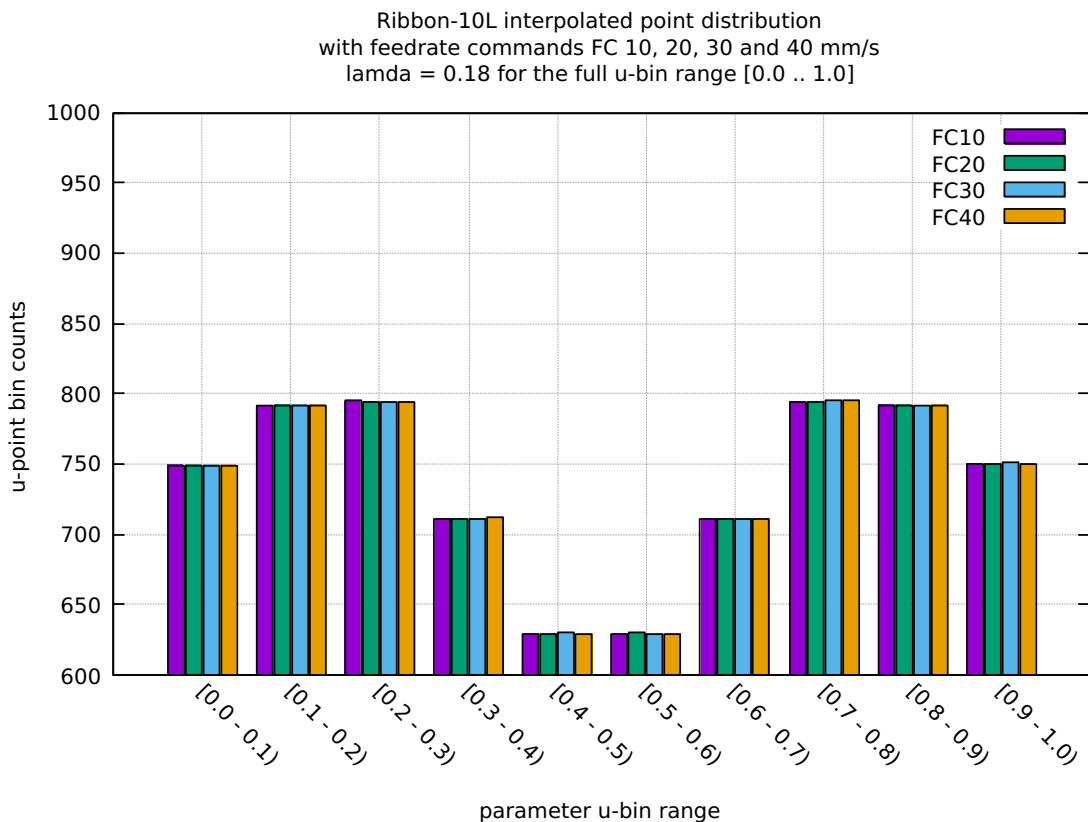


Table 14: Ribbon-10L Table distribution of interpolated points

BINS	FC10	FC20	FC30	FC40
0.0 - 0.1	749	749	749	749
0.1 - 0.2	791	792	792	792
0.2 - 0.3	795	794	794	794
0.3 - 0.4	711	711	711	712
0.4 - 0.5	629	629	630	629
0.5 - 0.6	629	630	629	629
0.6 - 0.7	711	711	711	711
0.7 - 0.8	794	794	795	795
0.8 - 0.9	792	792	791	792
0.9 - 1.0	750	750	751	750
Tot Counts	7351	7352	7353	7353

Table 15: Ribbon-10L Table FC10-20-30-40 Run Performance data

1	Curve Type	RIBBON-10L	RIBBON-10L	RIBBON-10L
2	User Feedrate Command FC(mm/s)	FC10 0.18	FC20 0.18	FC40 0.18
3	User Lamda Acceleration Safety Factor			
4	Total Interpolated Points (TIP)	7351	7352	7353
5	Total Sum-Chord-Error (SCE) (mm)	7.331686925442E-03	7.330650001960E-03	7.330843544266E-03
6	Ratio 1 = (SCE/TIP) = Chord-Error/Point	9.975084252302E-07	9.972316694273E-07	9.971223536814E-07
7	Total Sum-Arc-Length (SAL) (mm)	3.802699021307E+00	3.802672335504E+00	3.803478267930E+00
8	Total Sum-Chord-Length (SCL) (mm)	1.521079586306E+01	1.521068903483E+01	1.521191524702E+01
9	Difference = (SAL - SCL) (mm)	-1.140809684176E+01	-1.140801669932E+01	-1.140893646914E+01
10	Percentage Difference = (SAL - SCL)/SAL	-2.999999941577E+02	-2.999999919218E+02	-3.000000205782E+02
11	Ratio 2 = (SCE/SCL) = Chord Error/Chord-Length	4.820054776519E-04	4.819406921788E-04	4.818512589475E-04
12	Total Sum-Arc-Theta (SAT) (rad)	5.446619853644E+00	5.446616685362E+00	5.446717935311E+00
13	Total Sum-Arc-Area (SAA) (mm2)	1.338515095667E-06	1.338121185915E-06	1.338217831316E-06
14	Ratio 3 = (SAA/SCL) = Arc-Area/Chord-Length	4.820054776519E-04	4.819406921788E-04	4.818512589475E-04
15	Average-Chord-Error (ACE) (mm)	9.975084252302E-07	9.972316694273E-07	9.971223536814E-07
16	Average-Arc-Length (AAL) (mm)	5.173740165044E-04	5.173000048298E-04	5.173392638643E-04
17	Average-Chord-Length (ACL) (mm)	2.069496035791E-03	2.069199977531E-03	2.069357161916E-03
18	Average-Arc-Theta (AAT) (rad)	7.410367147815E-04	7.409354761749E-04	7.408484678062E-04
19	Average-Arc-Area (AAA) (mm2)	1.821108973696E-10	1.820325378744E-10	1.819805422838E-10
20	Algorithm actual runtime on computer (ART) (s)	65.683135834	59.539855092	75.452534184
				73.307374778

.10 APPENDIX RIBBON-100L CURVE

- .10.1 Plot of Ribbon-100L curve [260]**
- .10.2 Ribbon-100L Radius of Curvature [261]**
- .10.3 Ribbon-100L Validation in LinuxCNC [262]**
- .10.4 Ribbon-100L Direction of Travel 3D [263]**
- .10.5 Ribbon-100L First and Second Order Taylors App [264]**
- .10.6 Ribbon-100L First minus Second Order Taylors App [265]**
- .10.7 Ribbon-100L Separate First Second Order Taylors App [??]**
- .10.8 Ribbon-100L Separation SAL and SCL [267]**
- .10.9 Ribbon-100L Chord-error in close view 2 scales [268]**
- .10.10 Ribbon-100L Four Components Feedrate Limit [269]**
- .10.11 Ribbon-100L FrateCommand FrateLimit and Curr-Frate [270]**
- .10.12 Ribbon-100L FeedRateLimit minus CurrFeedRate [271]**
- .10.13 Ribbon-100L FC20-Nominal X and Y Feedrate Profiles [272]**
- .10.14 Ribbon-100L FC20 Nominal Tangential Acceleration [273]**
- .10.15 Ribbon-100L FC20 Nominal Rising S-Curve Profile [274]**
- .10.16 Ribbon-100L FC20 Nominal Falling S-Curve Profile [275]**
- .10.17 Ribbon-100L FC10 Colored Feedrate Profile ngcode [276]**
- .10.18 Ribbon-100L FC20 Colored Feedrate Profile ngcode [277]**

- .10.19 **Ribbon-100L FC30 Colored Feedrate Profile ngcode** [278]
- .10.20 **Ribbon-100L FC40 Colored Feedrate Profile ngcode** [279]
- .10.21 **Ribbon-100L FC10 Tangential Acceleration** [280]
- .10.22 **Ribbon-100L FC20 Tangential Acceleration** [281]
- .10.23 **Ribbon-100L FC30 Tangential Acceleration** [282]
- .10.24 **Ribbon-100L FC40 Tangential Acceleration** [283]
- .10.25 **Ribbon-100L FC20 Nominal Separation NAL and NCL** [284]
- .10.26 **Ribbon-100L SAL minus SCL FC10 FC20 FC30 FC40** [??]
- .10.27 **Ribbon-100L FC10 FrateCmd CurrFrate X-Frate Y-Frate** [286]
- .10.28 **Ribbon-100L FC20 FrateCmd CurrFrate X-Frate Y-Frate** [287]
- .10.29 **Ribbon-100L FC30 FrateCmd CurrFrate X-Frate Y-Frate** [288]
- .10.30 **Ribbon-100L FC40 FrateCmd CurrFrate X-Frate Y-Frate** [289]
- .10.31 **Ribbon-100L FC10 Four Components FeedrateLimit** [290]
- .10.32 **Ribbon-100L FC20 Four Components FeedrateLimit** [291]
- .10.33 **Ribbon-100L FC30 Four Components FeedrateLimit** [292]
- .10.34 **Ribbon-100L FC40 Four Components FeedrateLimit** [293]
- .10.35 **Ribbon-100L Histogram Points FC10 FC20 FC30 FC40** [294]
- .10.36 **Ribbon-100L Table distribution of interpolated points** [16]
- .10.37 **Ribbon-100L Table FC10-20-30-40 Run Performance data** [17]

Figure 260: Plot of Ribbon-100L curve

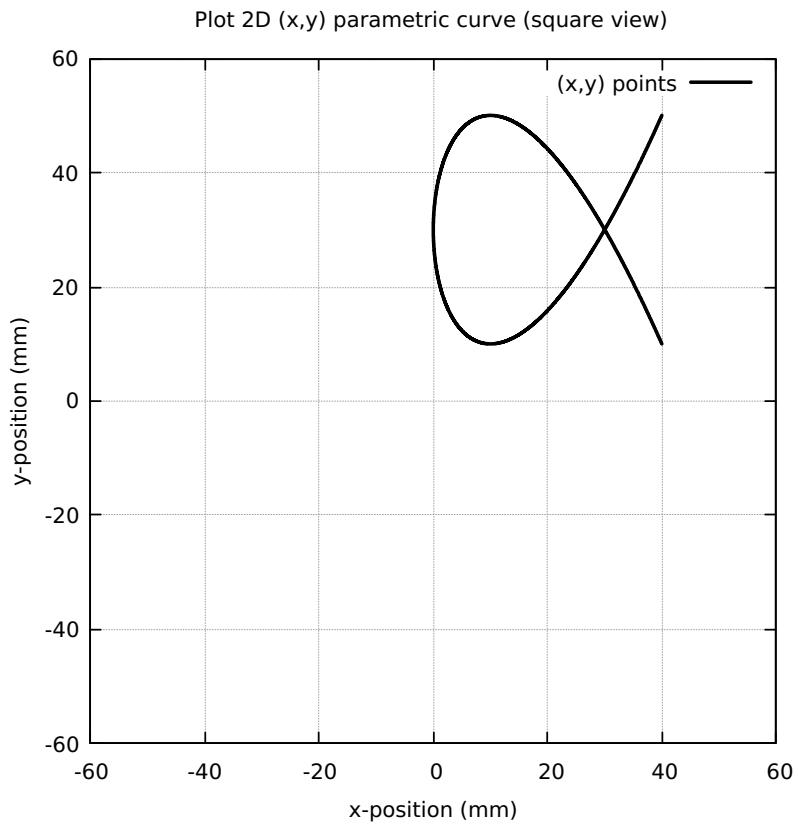


Figure 261: Ribbon-100L Radius of Curvature

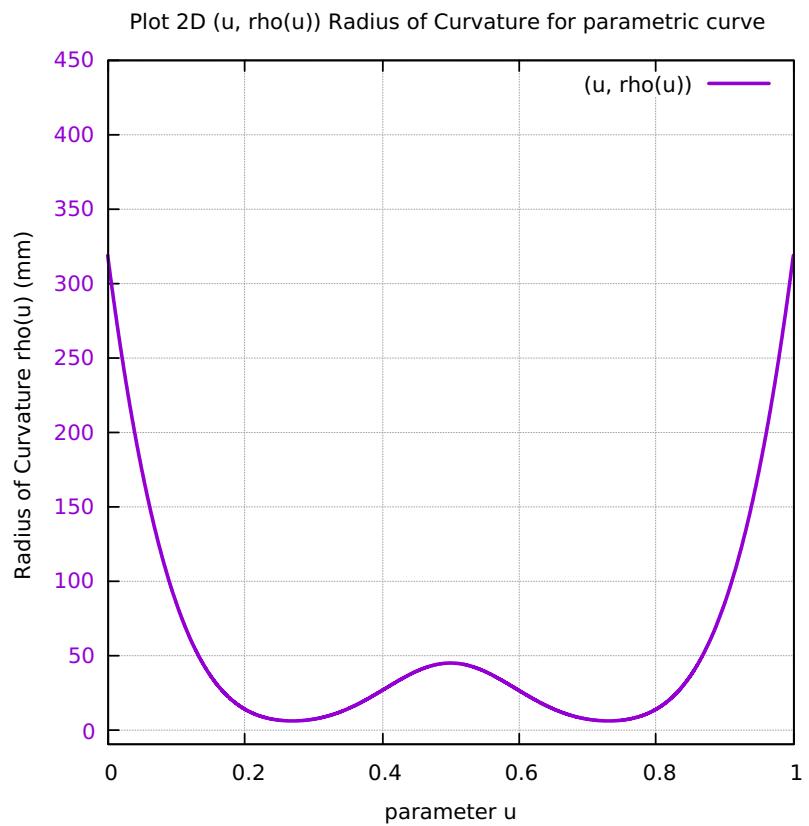


Figure 262: Ribbon-100L Validation in LinuxCNC

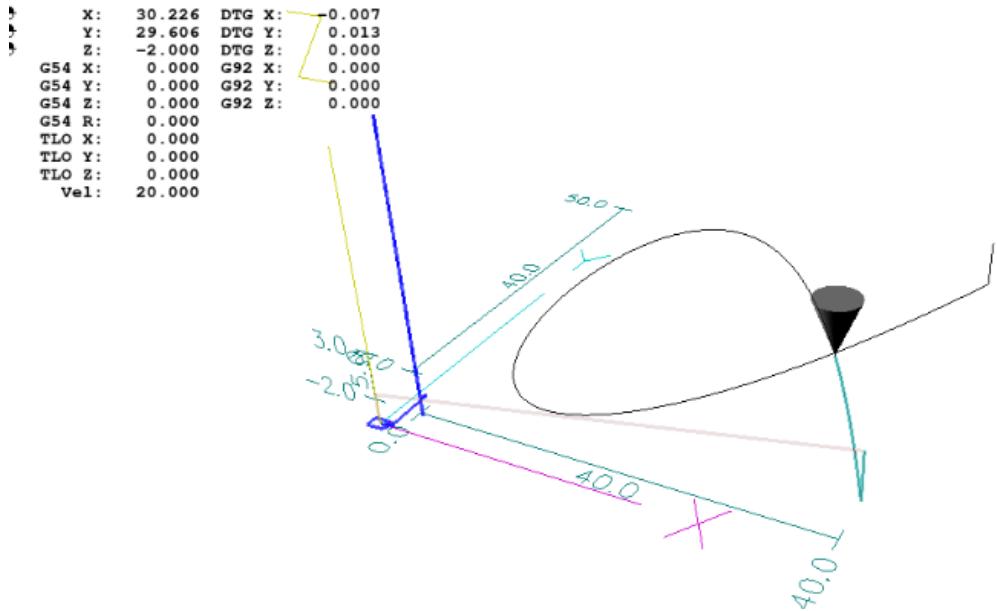


Figure 263: Ribbon-100L Direction of Travel 3D

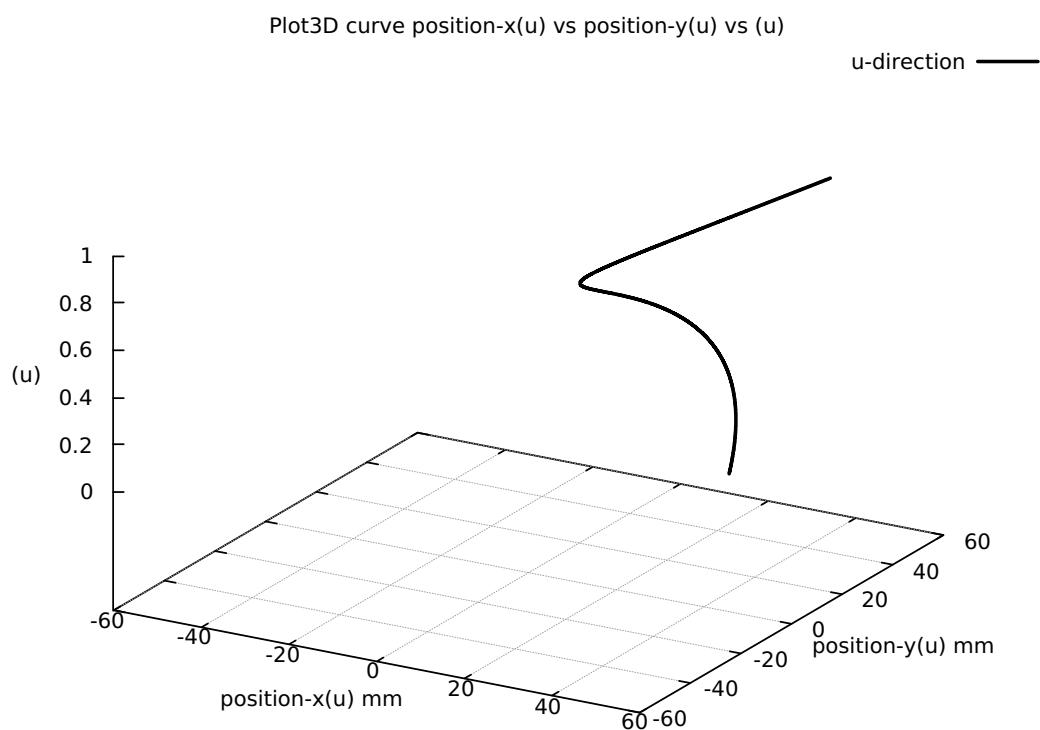


Figure 264: Ribbon-100L First and Second Order Taylor's Approximation

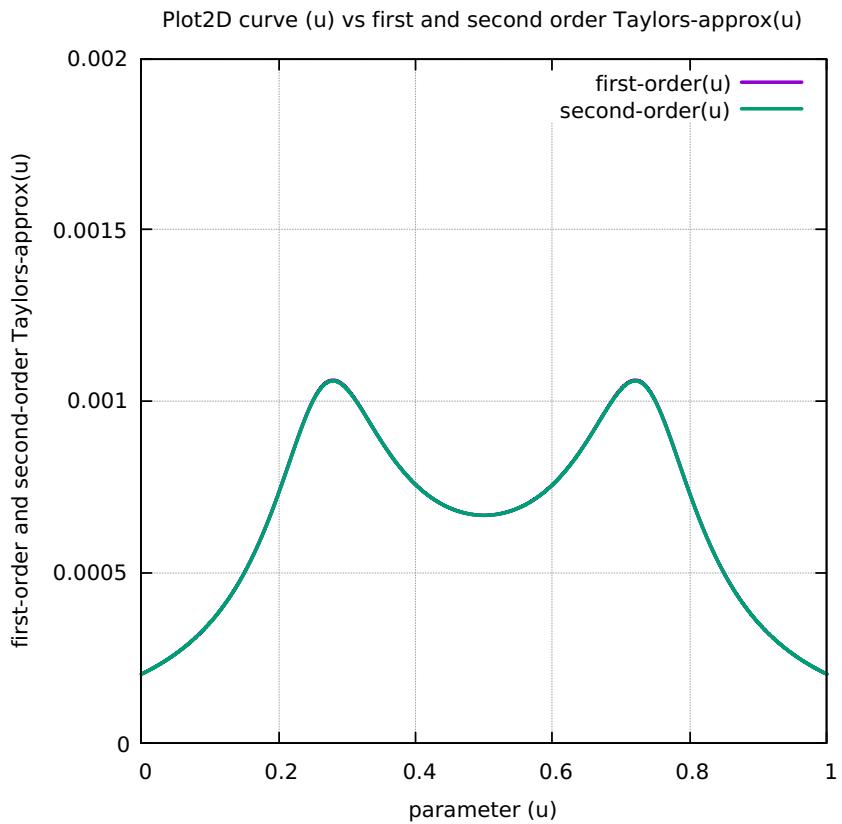


Figure 265: Ribbon-100L First minus Second Order Taylor's Approximation

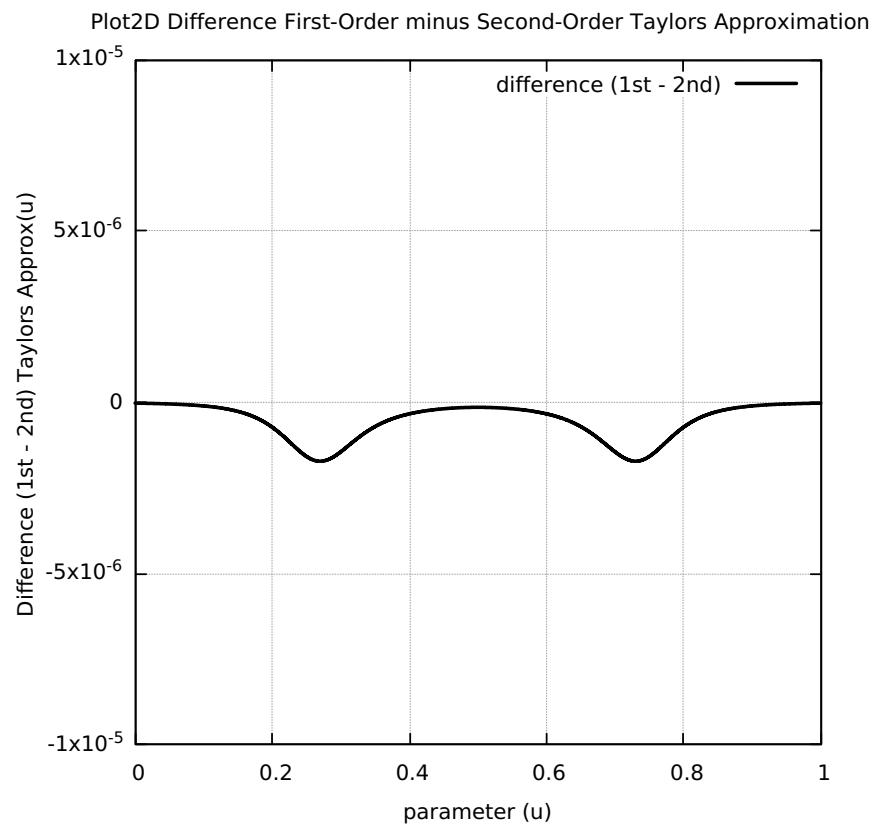


Figure 266: Ribbon-100L-First and Second Order Taylor's Approximation

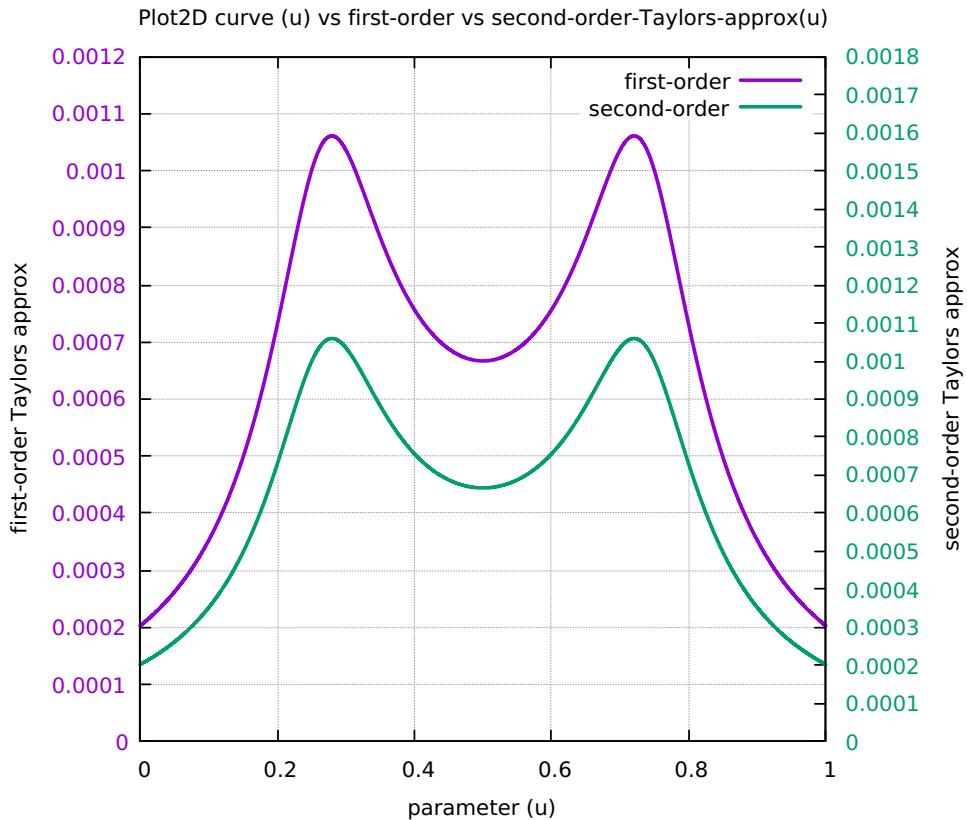


Figure 267: Ribbon-100L Separation SAL and SCL

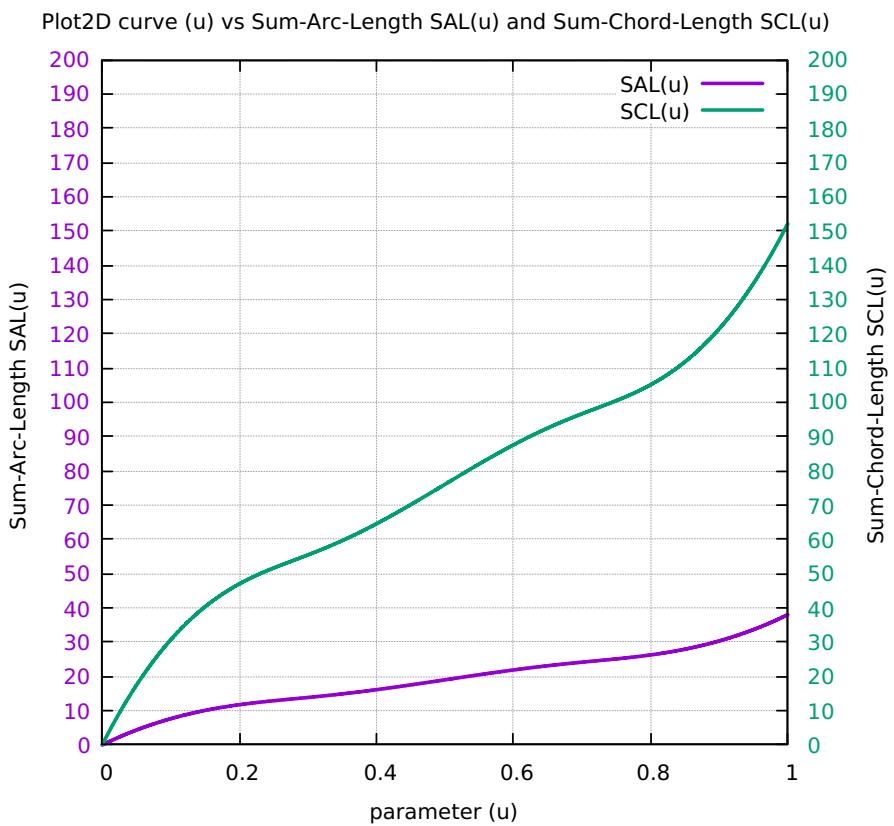


Figure 268: Ribbon-100L Chord-error in close view 2 scales

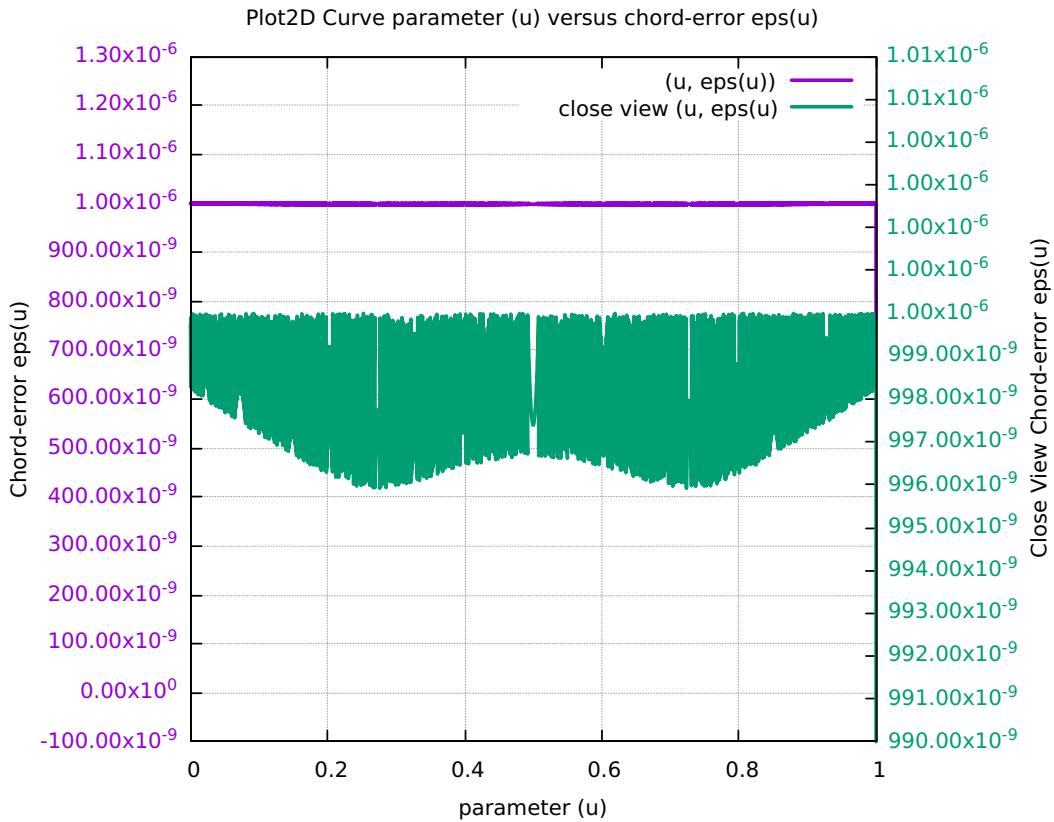


Figure 269: Ribbon-100L Four Components Feedrate Limit

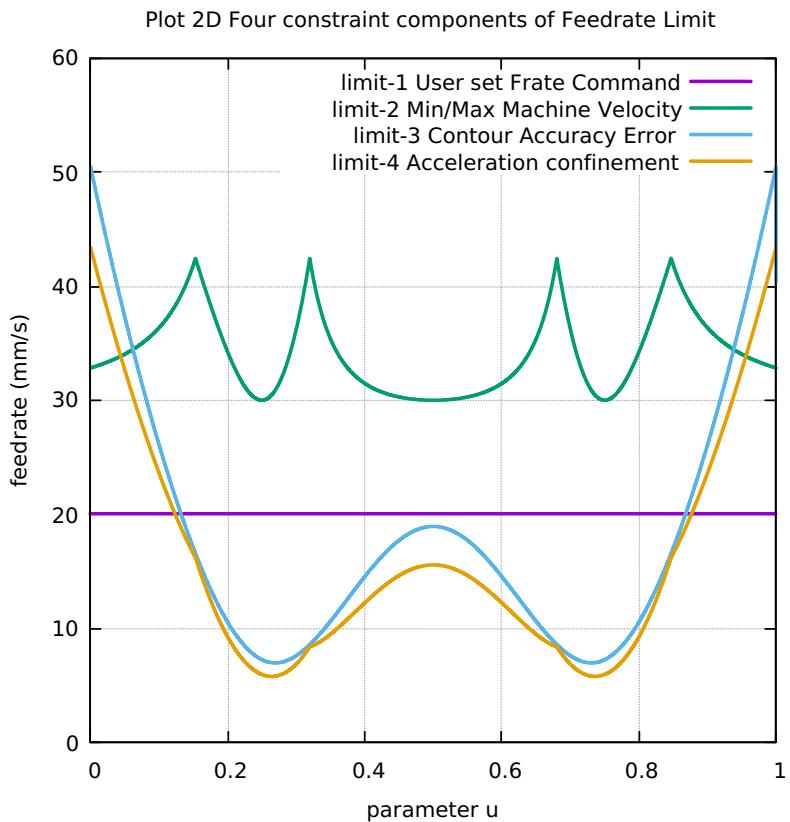


Figure 270: Ribbon-100L FrateCommand FrateLimit and Curr-Frate

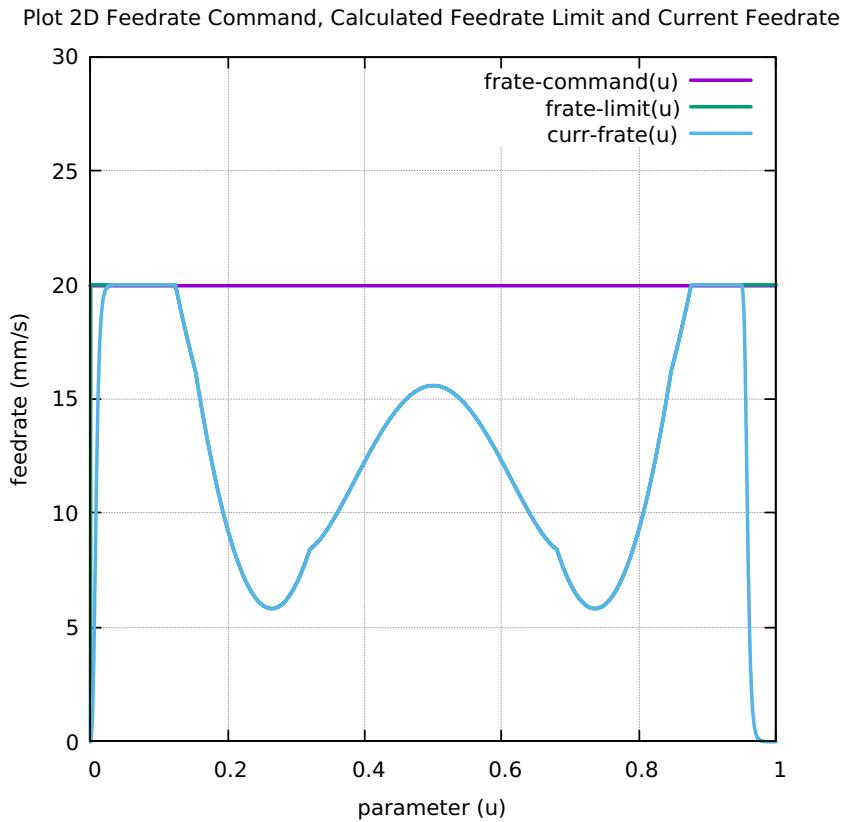


Figure 271: Ribbon-100L FeedRateLimit minus CurrFeedRate

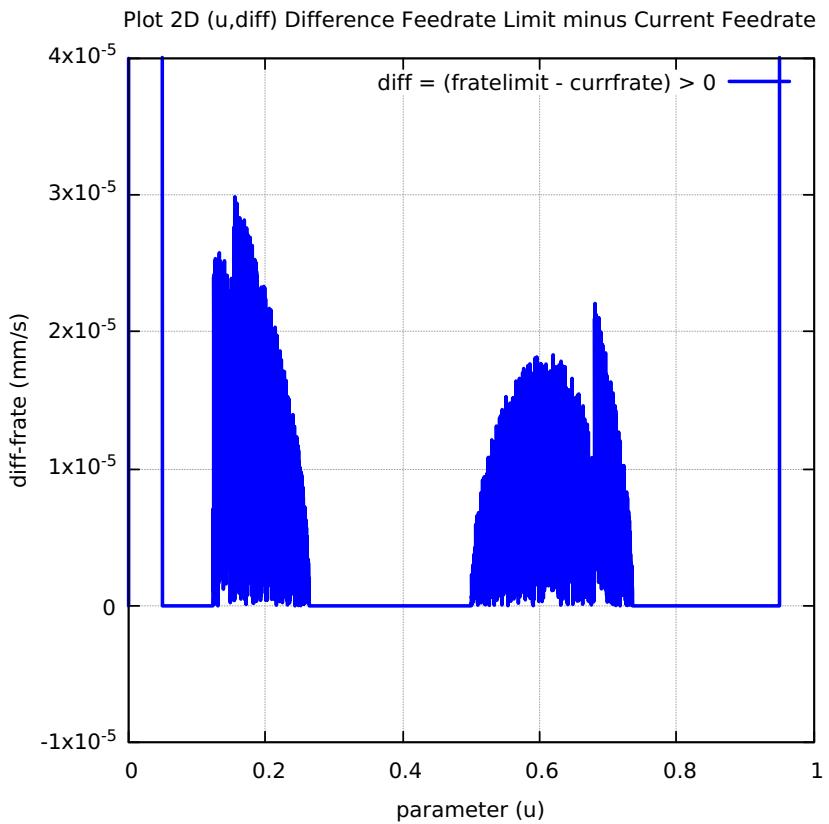


Figure 272: Ribbon-100L FC20-Nominal X and Y Feedrate Profiles

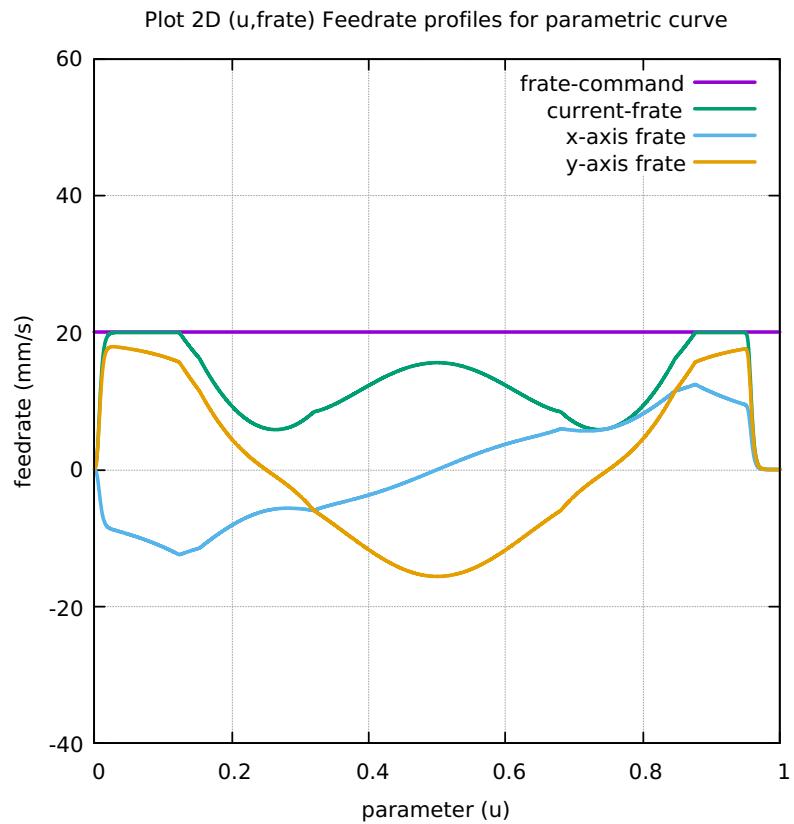


Figure 273: Ribbon-100L FC20 Nominal Tangential Acceleration

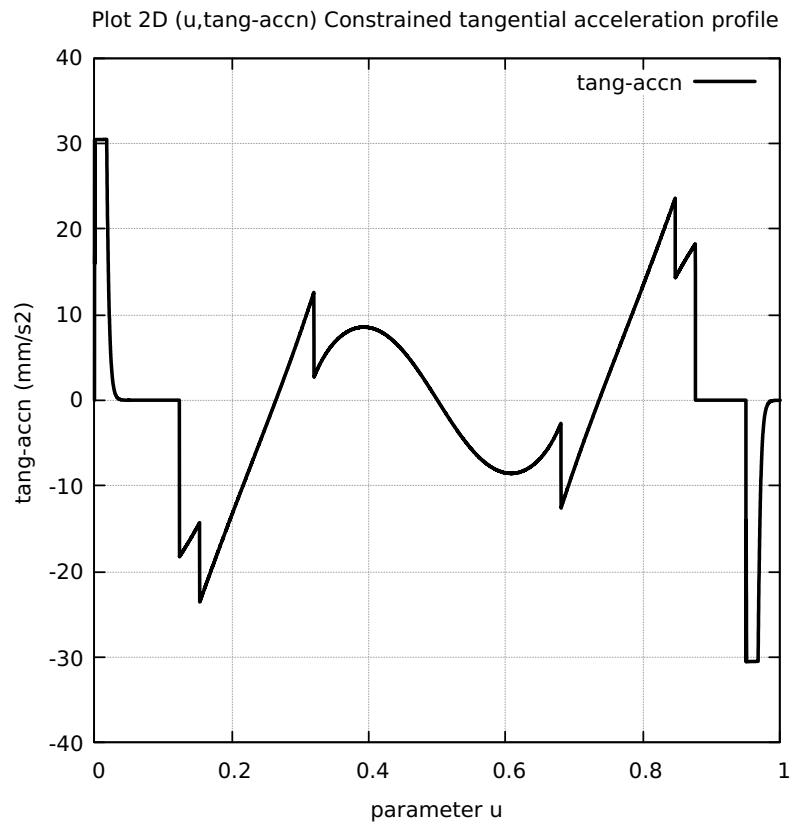


Figure 274: Ribbon-100L FC20 Nominal Rising S-Curve Profile

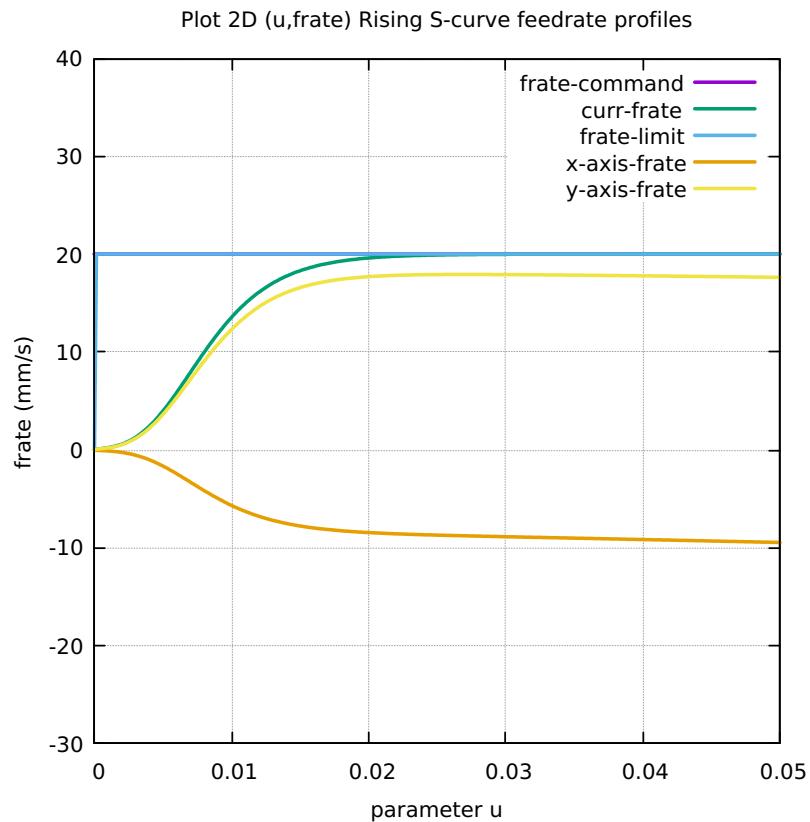


Figure 275: Ribbon-100L FC20 Nominal Falling S-Curve Profile

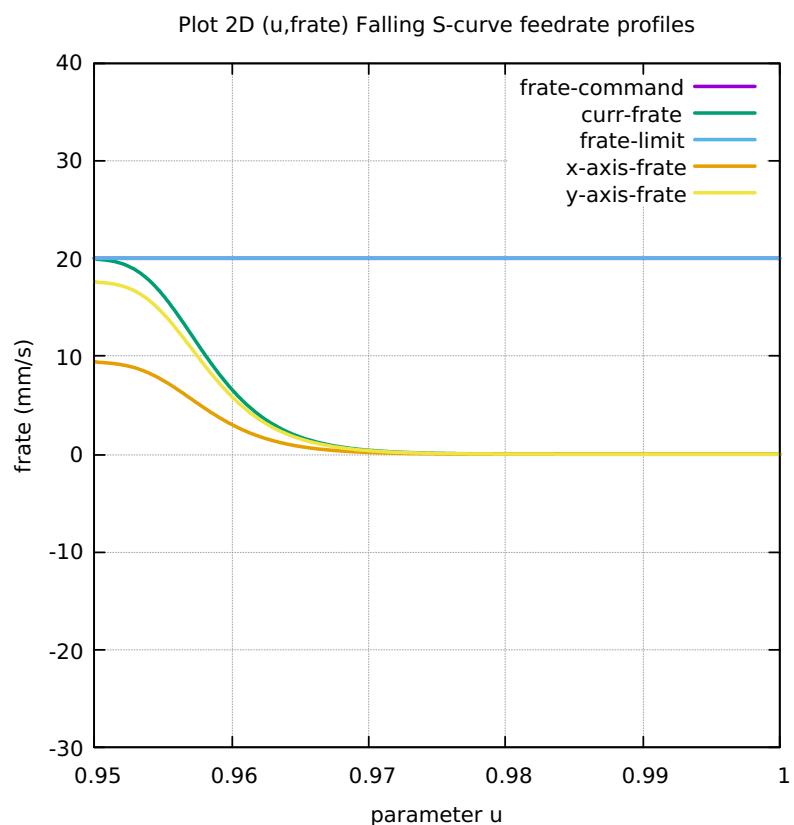


Figure 276: Ribbon-100L FC10 Colored Feedrate Profile data ngcode

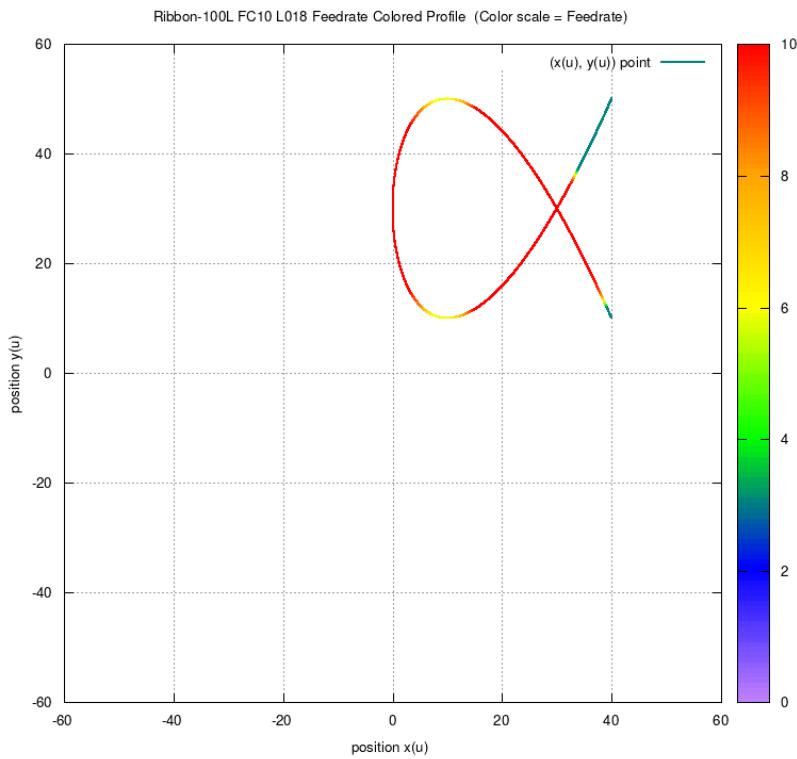


Figure 277: Ribbon-100L FC20 Colored Feedrate Profile data ngcode

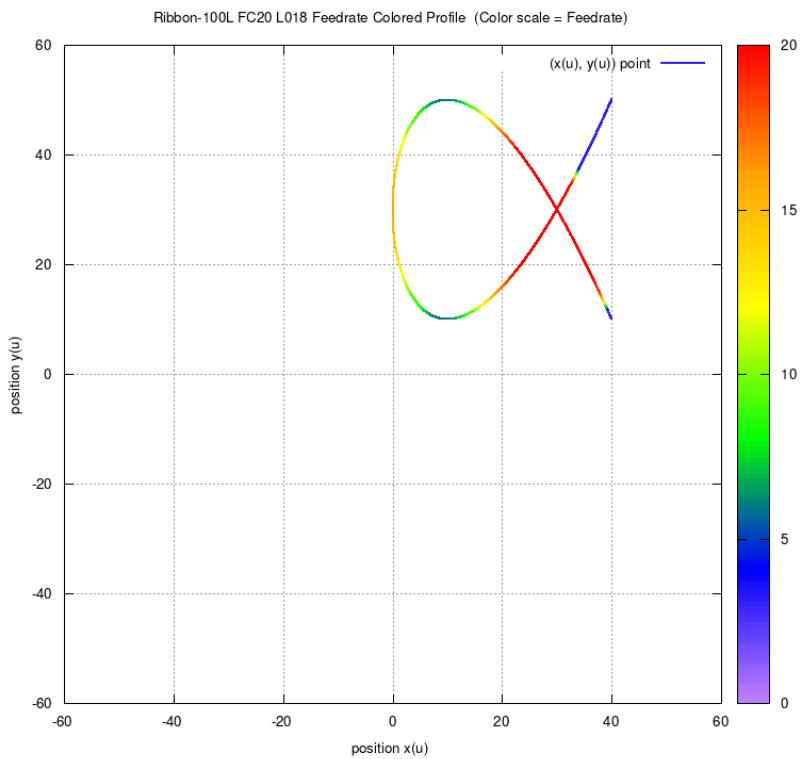


Figure 278: Ribbon-100L FC30 Colored Feedrate Profile data ngcode

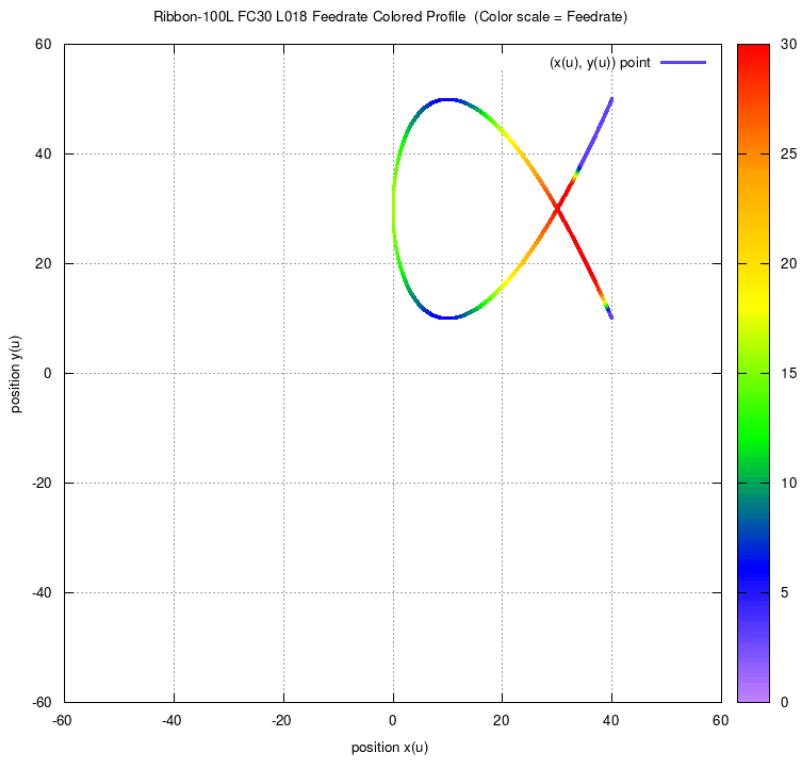


Figure 279: Ribbon-100L FC40 Colored Feedrate Profile data ngcode

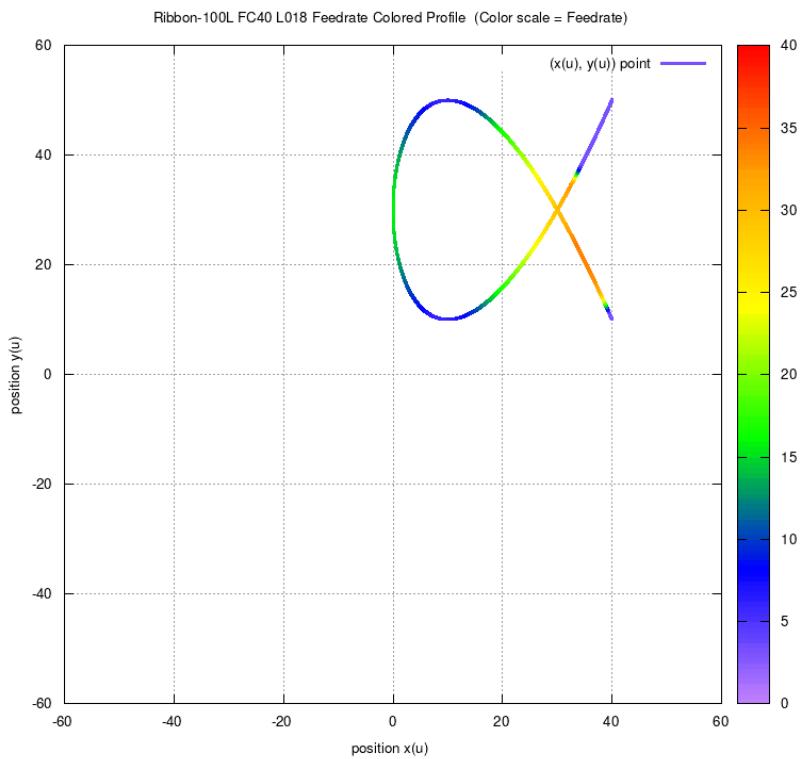


Figure 280: Ribbon-100L FC10 Tangential Acceleration

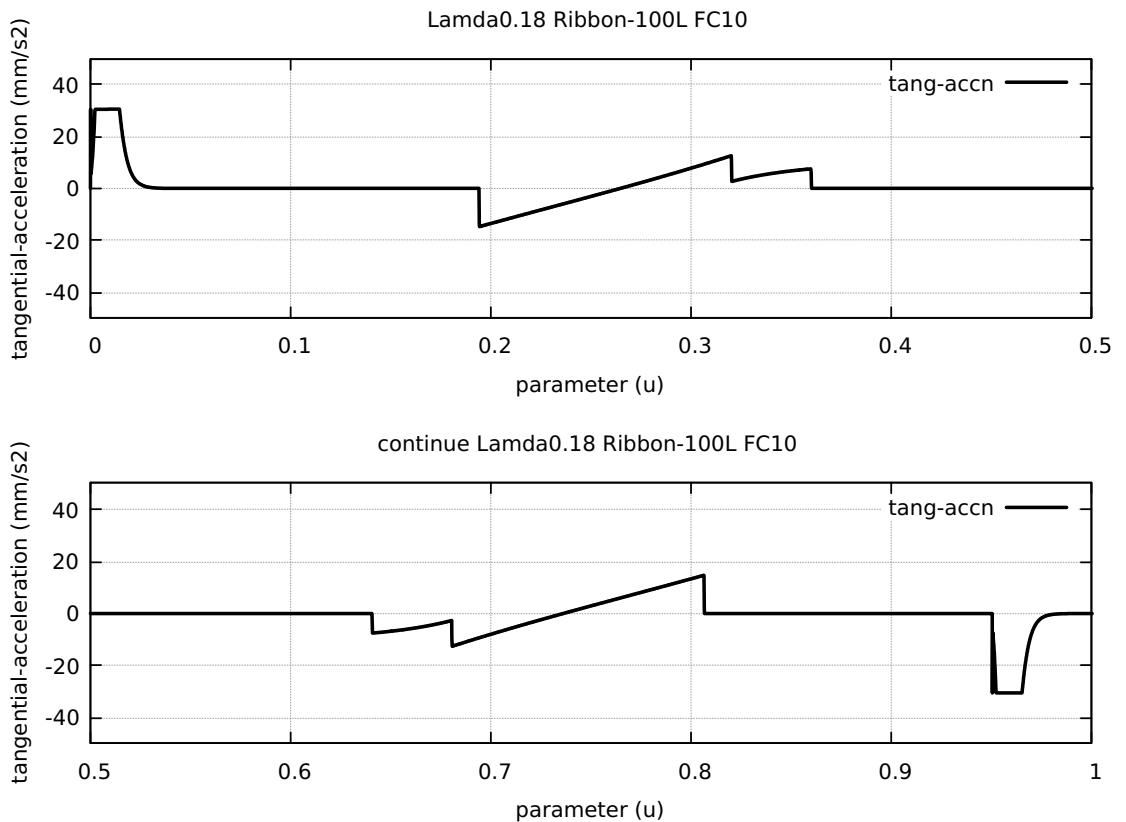


Figure 281: Ribbon-100L FC20 Tangential Acceleration

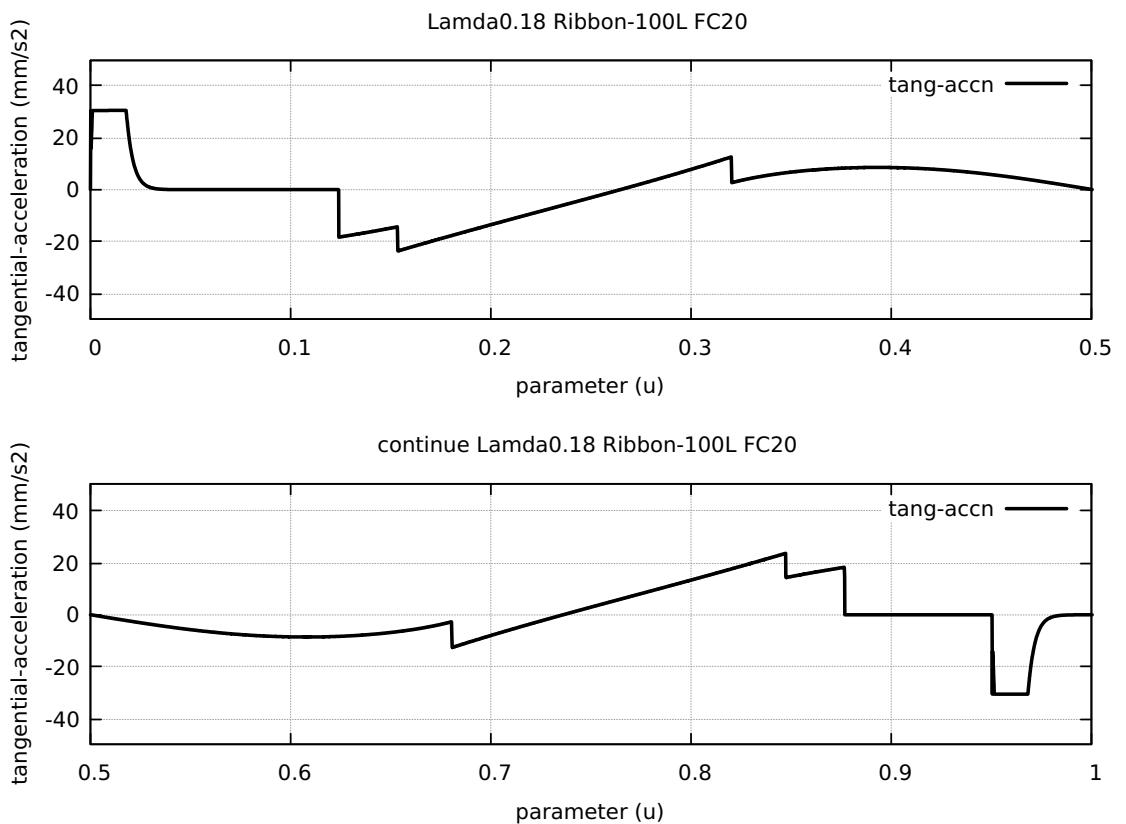


Figure 282: Ribbon-100L FC30 Tangential Acceleration

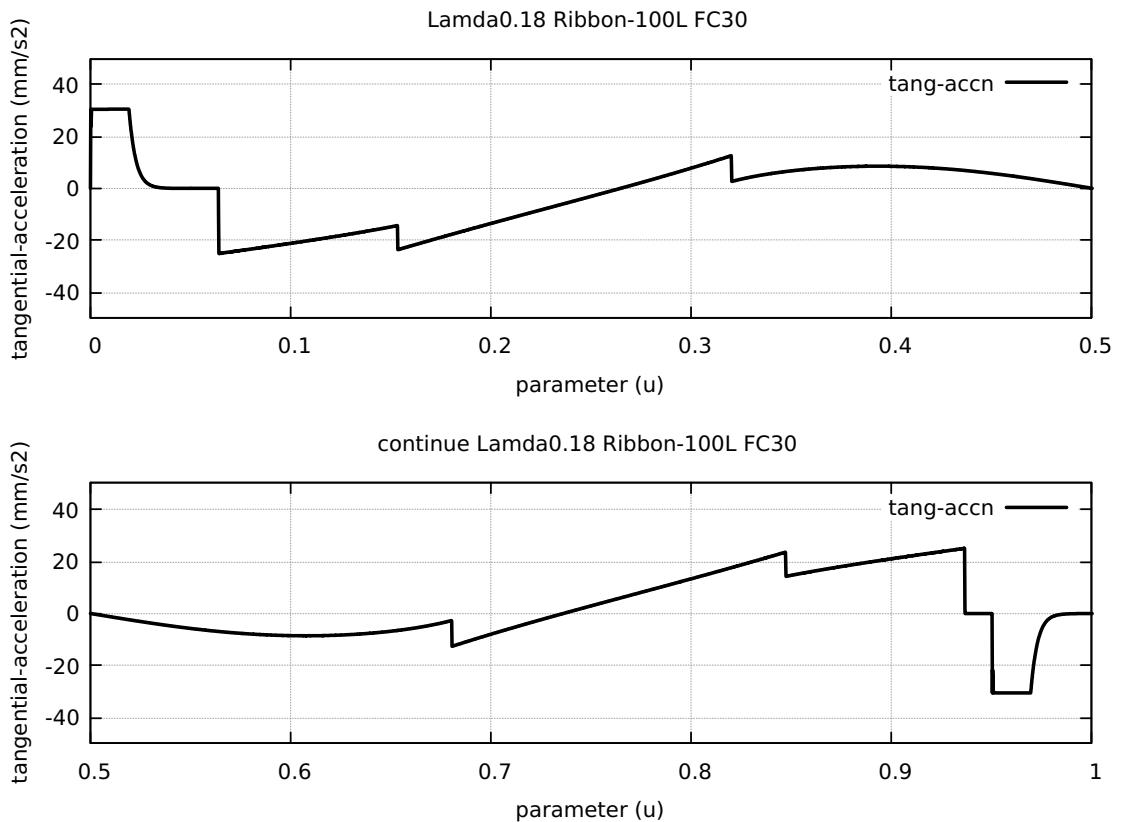


Figure 283: Ribbon-100L FC40 Tangential Acceleration

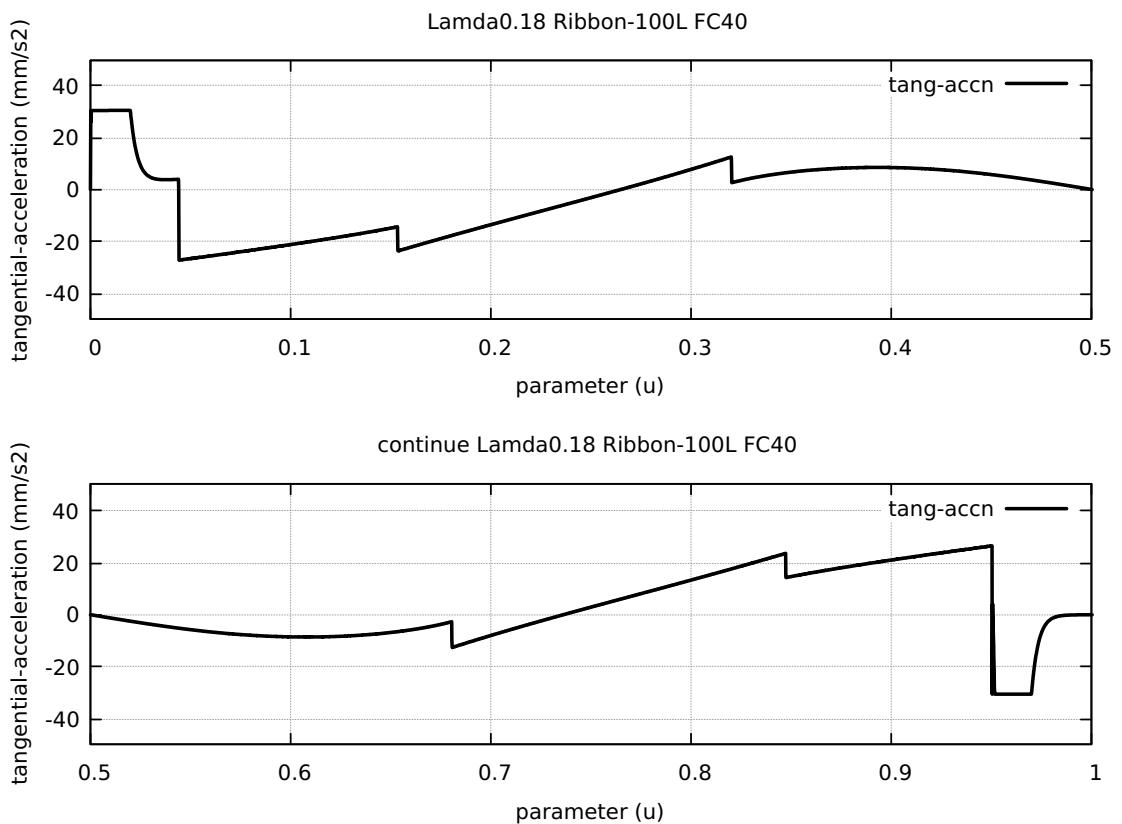


Figure 284: Ribbon-100L FC20 Nominal Separation NAL and NCL

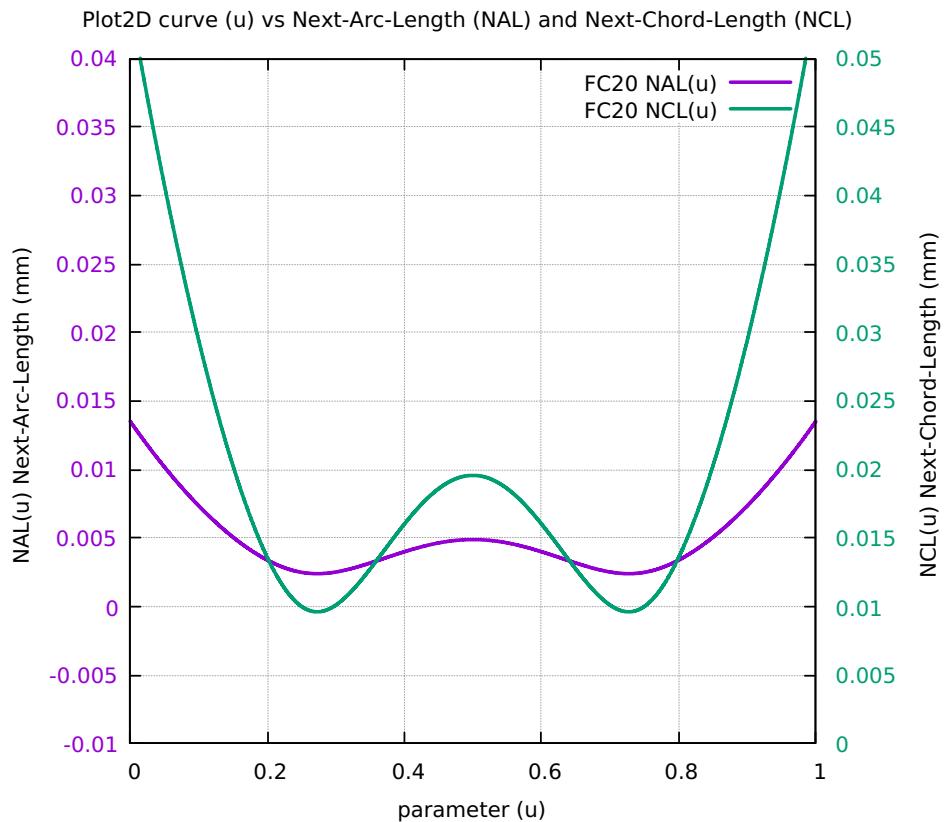


Figure 285: Ribbon-100L SAL minus SCL for FC10 FC20 FC30 FC40

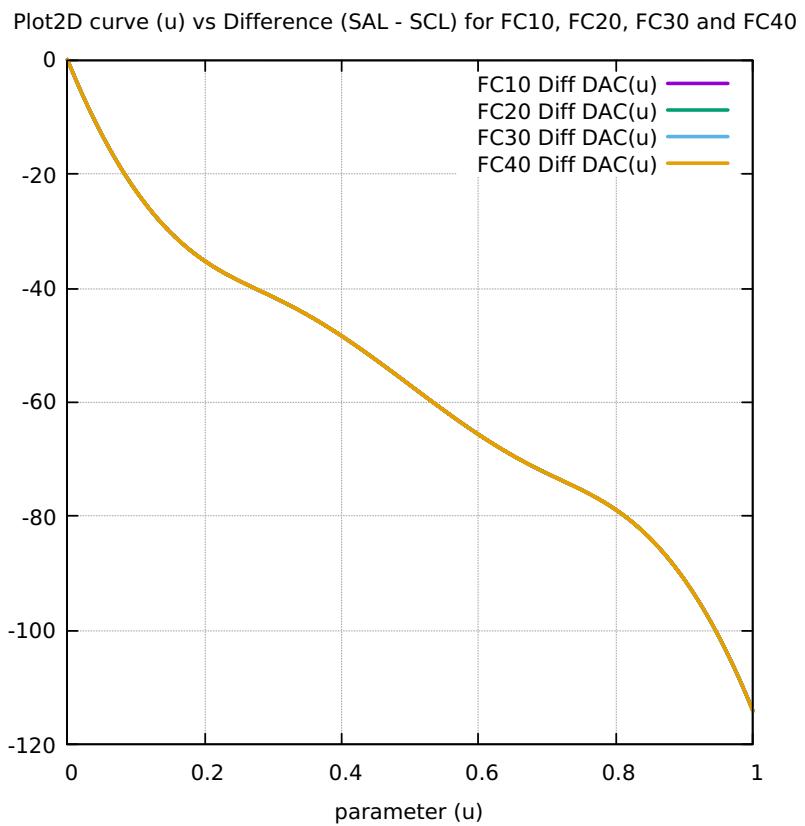


Figure 286: Ribbon-100L FC10 FrateCmd CurrFrate X-Frate Y-Frate

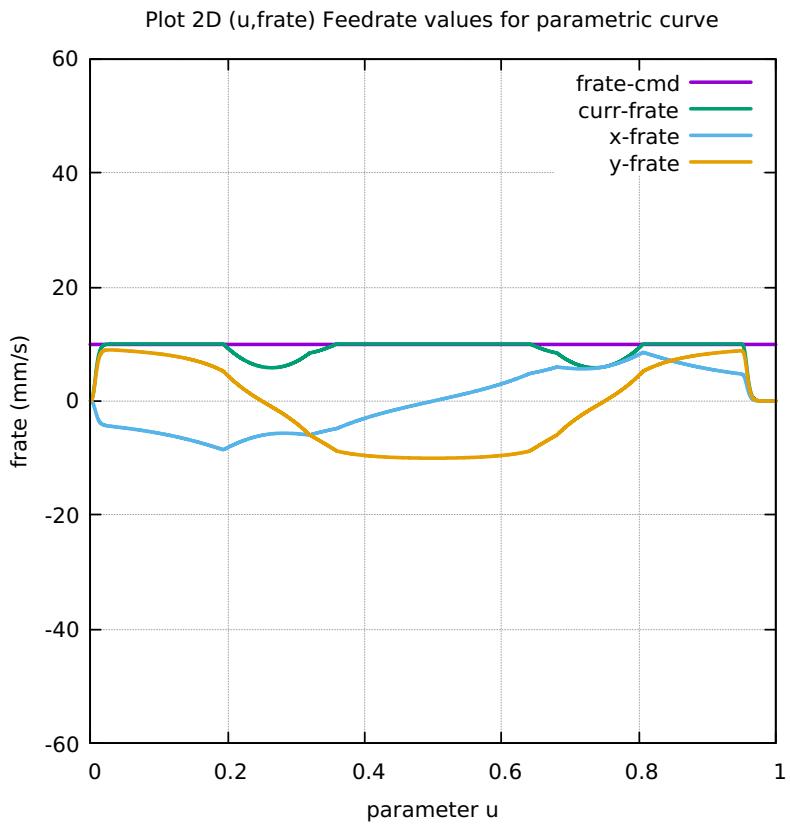


Figure 287: Ribbon-100L FC20 FrateCmd CurrFrate X-Frate Y-Frate

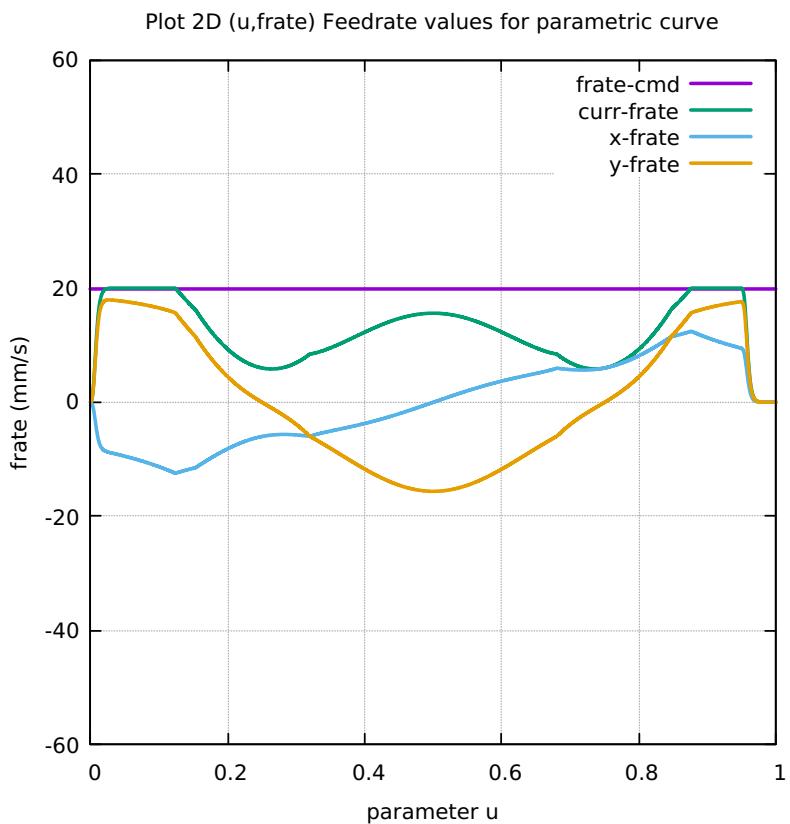


Figure 288: Ribbon-100L FC30 FrateCmd CurrFrate X-Frate Y-Frate

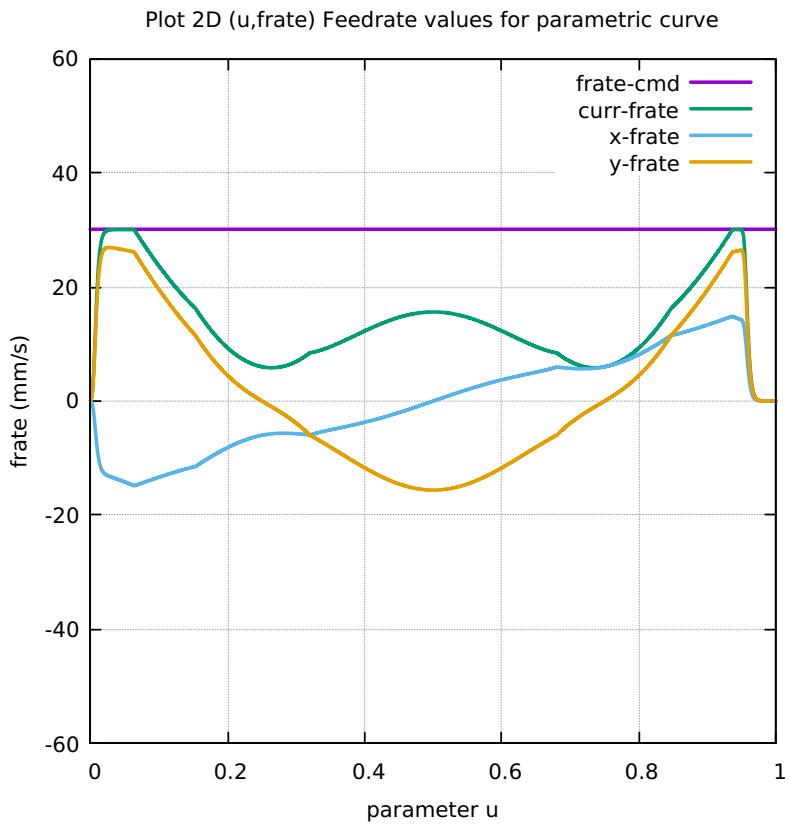


Figure 289: Ribbon-100L FC40 FrateCmd CurrFrate X-Frate Y-Frate

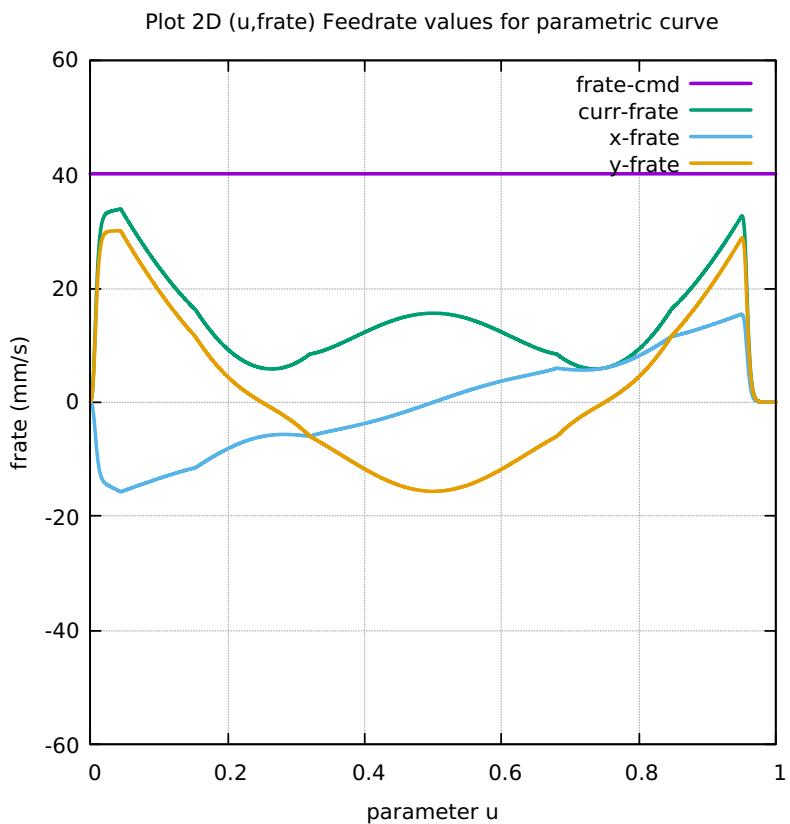


Figure 290: Ribbon-100L FC10 Four Components FeedrateLimit

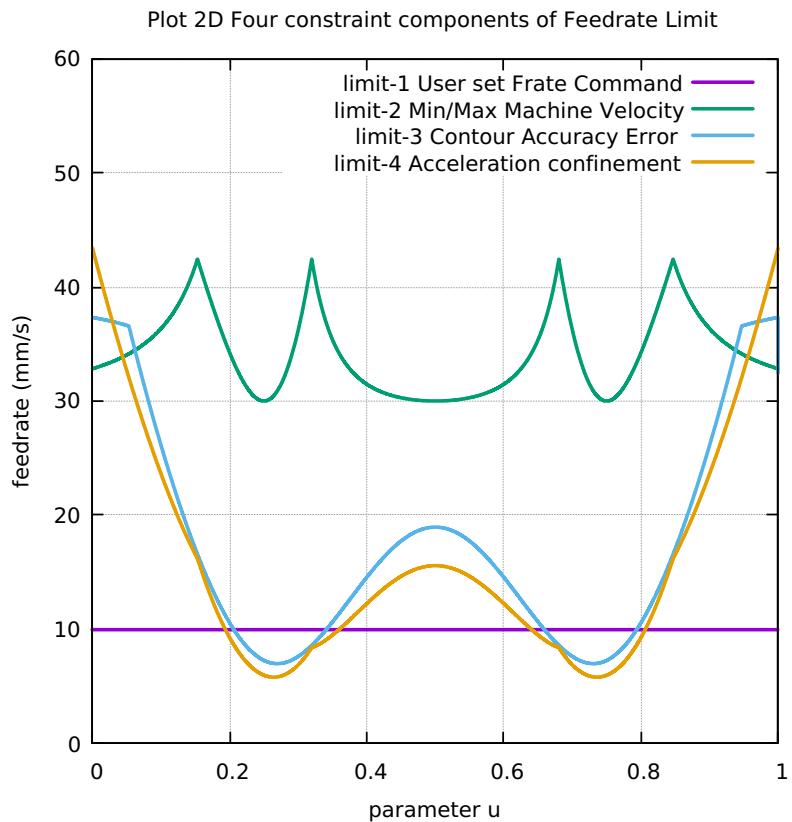


Figure 291: Ribbon-100L FC20 Four Components FeedrateLimit

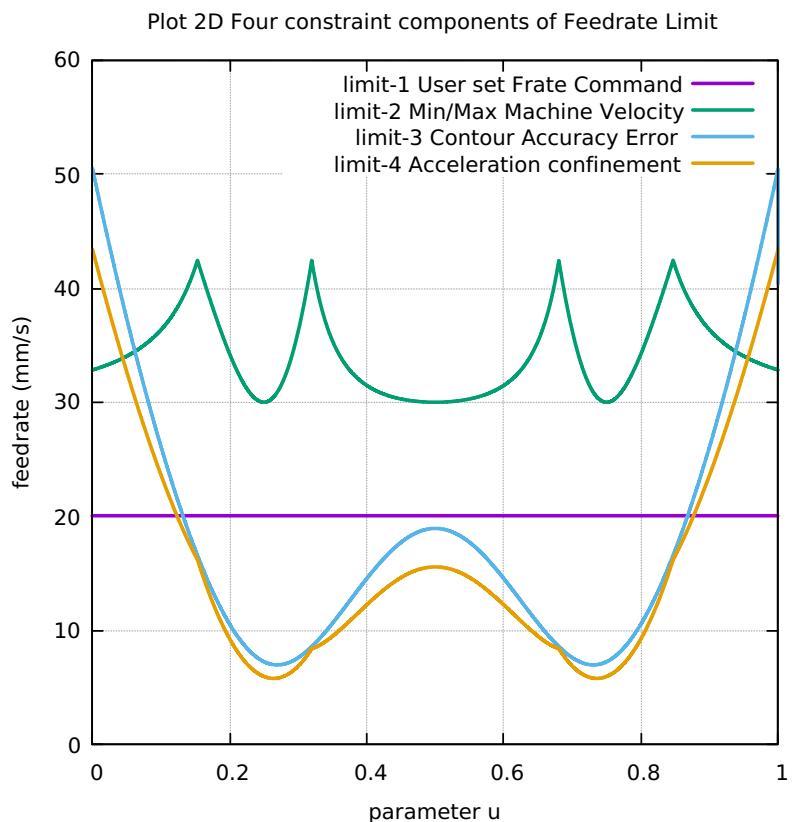


Figure 292: Ribbon-100L FC30 Four Components FeedrateLimit

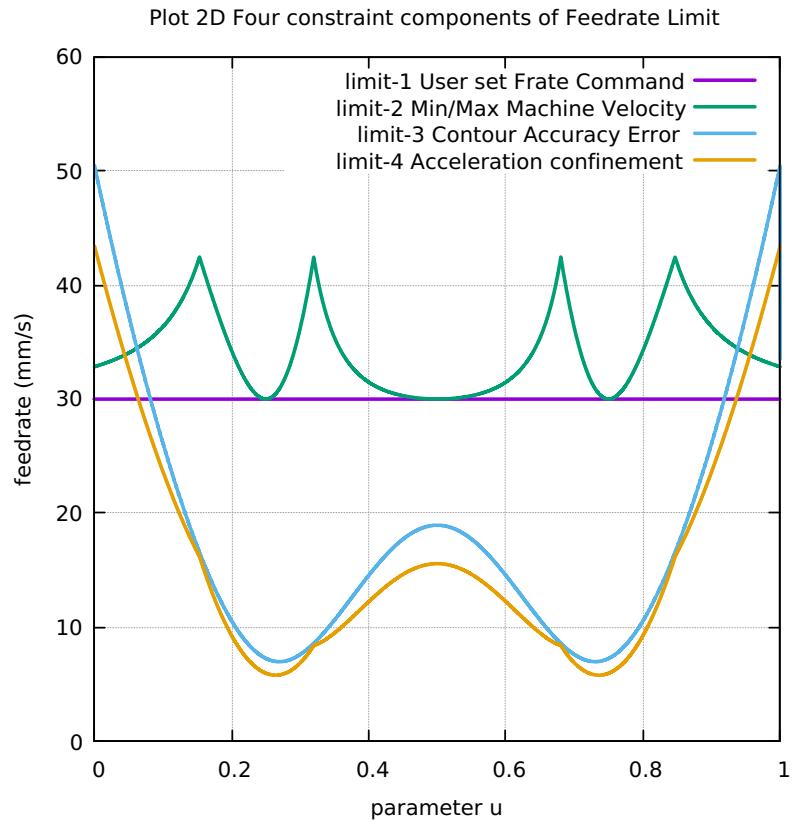


Figure 293: Ribbon-100L FC40 Four Components FeedrateLimit

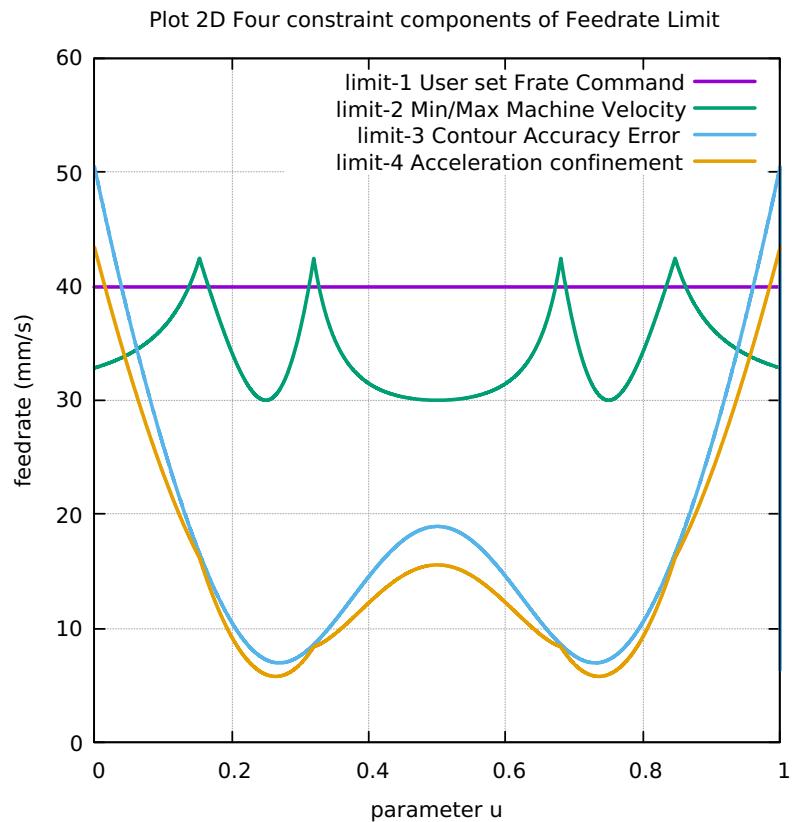


Figure 294: Ribbon-100L Histogram Points FC10 FC20 FC30 FC40

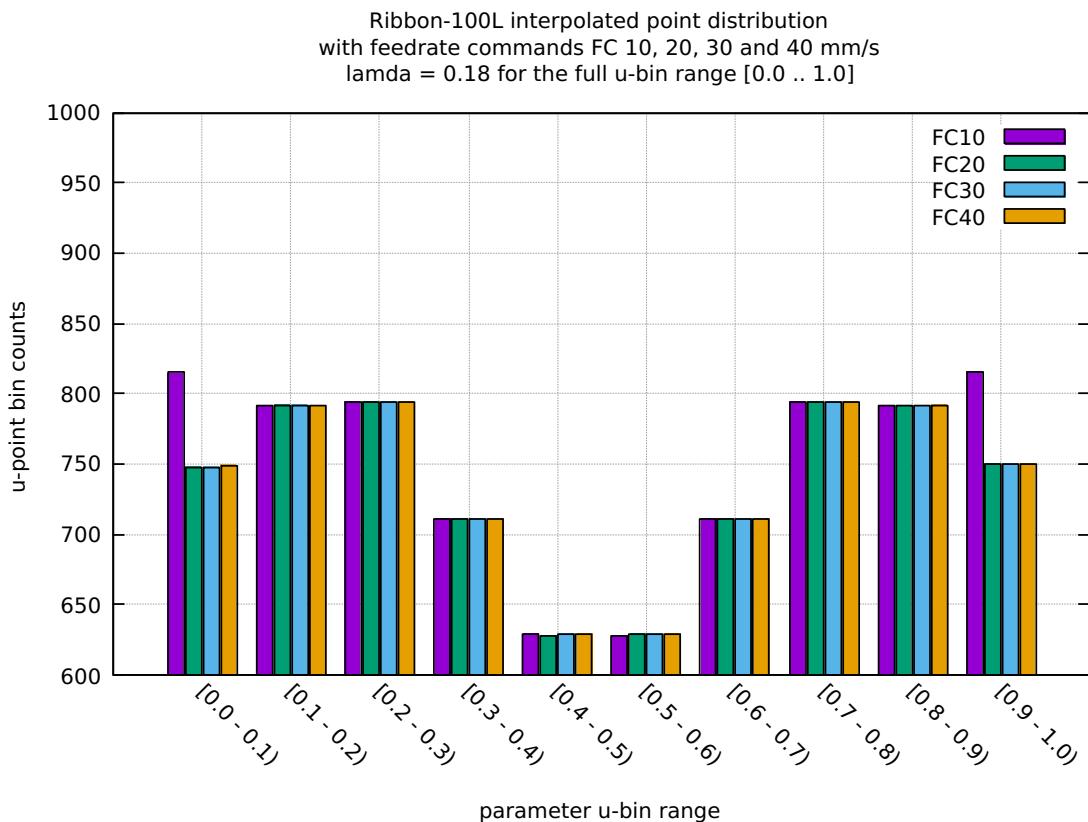


Table 16: Ribbon-100L Table distribution of interpolated points

BINS	FC10	FC20	FC30	FC40
0.0 - 0.1	815	748	748	749
0.1 - 0.2	791	792	792	791
0.2 - 0.3	794	794	794	794
0.3 - 0.4	711	711	711	711
0.4 - 0.5	629	628	629	629
0.5 - 0.6	628	629	629	629
0.6 - 0.7	711	711	711	711
0.7 - 0.8	794	794	794	794
0.8 - 0.9	791	791	791	792
0.9 - 1.0	816	750	750	750
Tot Counts	7480	7348	7349	7350

Table 17: Ribbon-100L Table FC10-20-30-40 Run Performance data

1	Curve Type	RIBBON-100L	RIBBON-100L	RIBBON-100L
2	User Feedrate Command FC(mm/s)	FC10 0.18	FC20 0.18	FC40 0.18
3	User Lamda Acceleration Safety Factor			
4	Total Interpolated Points (TIP)	7480	7348	7349
5	Total Sum-Chord-Error (SCE) (mm)	7.227413654822E-03	7.334488978808E-03	7.333764811636E-03
6	Ratio 1 = (SCE/TIP) = Chord-Error/Point	9.663609646773E-07	9.982971251951E-07	9.980627125254E-07
7	Total Sum-Arc-Length (SAL) (mm)	3.802433940498E+01	3.802572368417E+01	3.802757758292E+01
8	Total Sum-Chord-Length (SCL) (mm)	1.520973548479E+02	1.521028906780E+02	1.521393532475E+02
9	Difference = (SAL - SCL) (mm)	-1.140730154429E+02	-1.140771669938E+02	-1.141045169238E+02
10	Percentage Difference = (SAL - SCL)/SAL	-2.999999927099E+02	-2.999999893265E+02	-2.99999955199E+02
11	Ratio 2 = (SCE/SCL) = Chord Error/Chord-Length	4.751833890898E-05	4.822057586226E-05	4.821346349097E-05
12	Total Sum-Arc-Theta (SAT) (rad)	5.446630718238E+00	5.446603832206E+00	5.446627300213E+00
13	Total Sum-Arc-Area (SAA) (mm2)	1.265139232664E-04	1.339597112819E-04	1.339309239988E-04
14	Ratio 3 = (SAA/SCL) = Arc-Area/Chord-Length	4.751833890898E-05	4.822057586226E-05	4.821346349097E-05
15	Average-Chord-Error (ACE) (mm)	9.663609646773E-07	9.982971251951E-07	9.980627125254E-07
16	Average-Arc-Length (AAL) (mm)	5.084147533759E-03	5.175680370787E-03	5.175228304698E-03
17	Average-Chord-Length (ACL) (mm)	2.033658976439E-02	2.070272093072E-02	2.070091298694E-02
18	Average-Arc-Theta (AAT) (rad)	7.282565474312E-04	7.413371215743E-04	7.412394257230E-04
19	Average-Arc-Area (AAA) (mm2)	1.691588758743E-08	1.823325320292E-08	1.822685410980E-08
20	Algorithm actual runtime on computer (ART) (s)	26.306206937	37.694118593	42.698734839
				47.339581162

.11 APPENDIX ASTEPI CURVE

- .11.1 Plot of AstEpi curve [295]**
- .11.2 AstEpi Radius of Curvature [296]**
- .11.3 AstEpi Validation in LinuxCNC [297]**
- .11.4 AstEpi Direction of Travel 3D [298]**
- .11.5 AstEpi First and Second Order Taylors App [299]**
- .11.6 AstEpi First minus Second Order Taylors App [300]**
- .11.7 AstEpi Separate First Second Order Taylors App [301]**
- .11.8 AstEpi Separation SAL and SCL [302]**
- .11.9 AstEpi Chord-error in close view 2 scales [303]**
- .11.10 AstEpi Four Components Feedrate Limit [304]**
- .11.11 AstEpi FrateCommand FrateLimit and Curr-Frate [305]**
- .11.12 AstEpi FeedRateLimit minus CurrFeedRate [306]**
- .11.13 AstEpi FC20-Nominal X and Y Feedrate Profiles [307]**
- .11.14 AstEpi FC20 Nominal Tangential Acceleration [308]**
- .11.15 AstEpi FC20 Nominal Rising S-Curve Profile [309]**
- .11.16 AstEpi FC20 Nominal Falling S-Curve Profile [310]**
- .11.17 AstEpi FC10 Colored Feedrate Profile ngcode [311]**
- .11.18 AstEpi FC20 Colored Feedrate Profile ngcode [312]**

- .11.19 AstEpi FC30 Colored Feedrate Profile ngcode [313]
- .11.20 AstEpi FC40 Colored Feedrate Profile ngcode [314]
- .11.21 AstEpi FC10 Tangential Acceleration [315]
- .11.22 AstEpi FC20 Tangential Acceleration [316]
- .11.23 AstEpi FC30 Tangential Acceleration [317]
- .11.24 AstEpi FC40 Tangential Acceleration [318]
- .11.25 AstEpi FC20 Nominal Separation NAL and NCL [319]
- .11.26 AstEpi SAL minus SCL for FC10 FC20 FC30 FC40 [320]
- .11.27 AstEpi FC10 FrateCmd CurrFrate X-Frate Y-Frate [321]
- .11.28 AstEpi FC20 FrateCmd CurrFrate X-Frate Y-Frate [322]
- .11.29 AstEpi FC30 FrateCmd CurrFrate X-Frate Y-Frate [323]
- .11.30 AstEpi FC40 FrateCmd CurrFrate X-Frate Y-Frate [324]
- .11.31 AstEpi FC10 Four Components FeedrateLimit [325]
- .11.32 AstEpi FC20 Four Components FeedrateLimit [326]
- .11.33 AstEpi FC30 Four Components FeedrateLimit [327]
- .11.34 AstEpi FC40 Four Components FeedrateLimit [328]
- .11.35 AstEpi Histogram Points FC10 FC20 FC30 FC40 [329]
- .11.36 AstEpi Table distribution of interpolated points [18]
- .11.37 AstEpi Table FC10-20-30-40 Run Performance data [19]

Figure 295: Plot of AstEpi curve

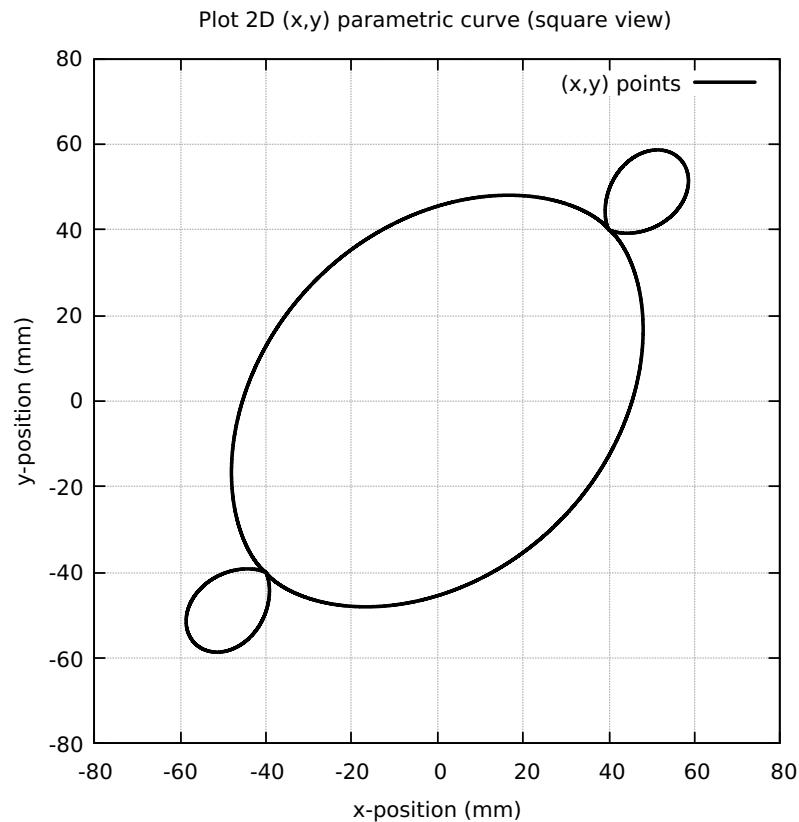


Figure 296: AstEpi Radius of Curvature

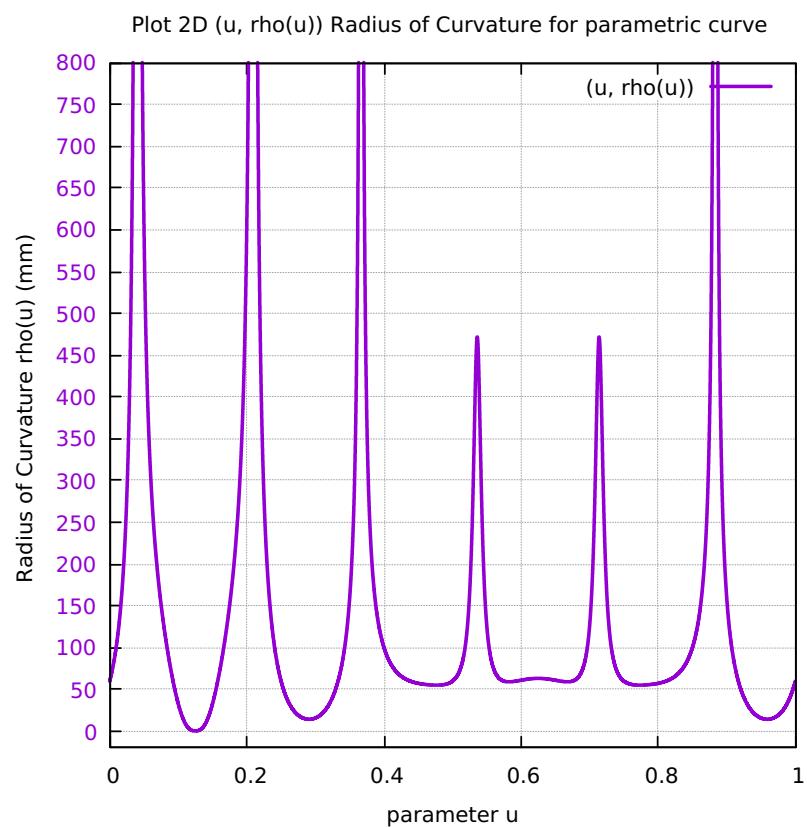


Figure 297: AstEpi Validation in LinuxCNC

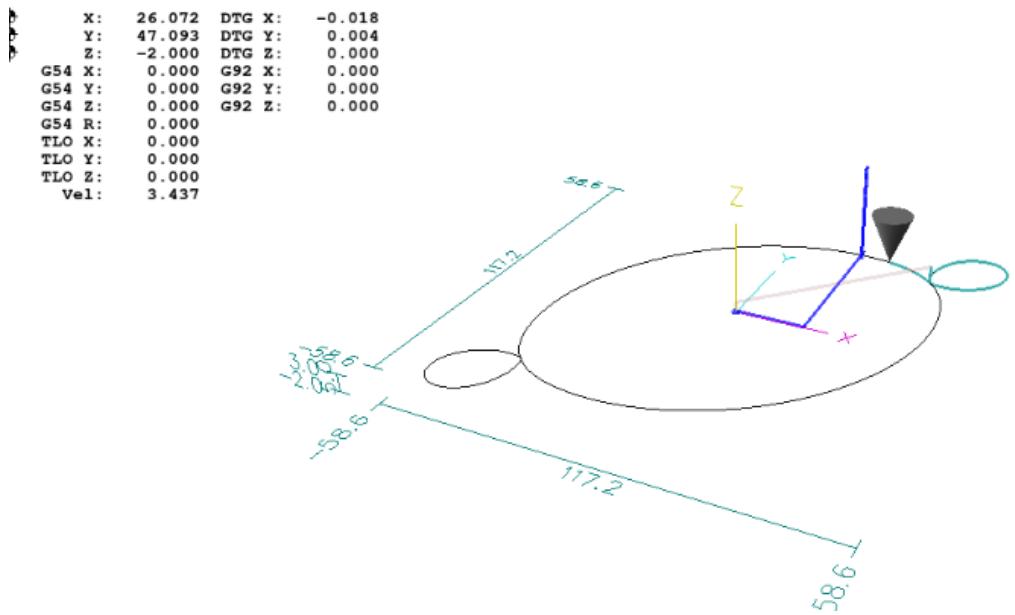


Figure 298: AstEpi Direction of Travel 3D

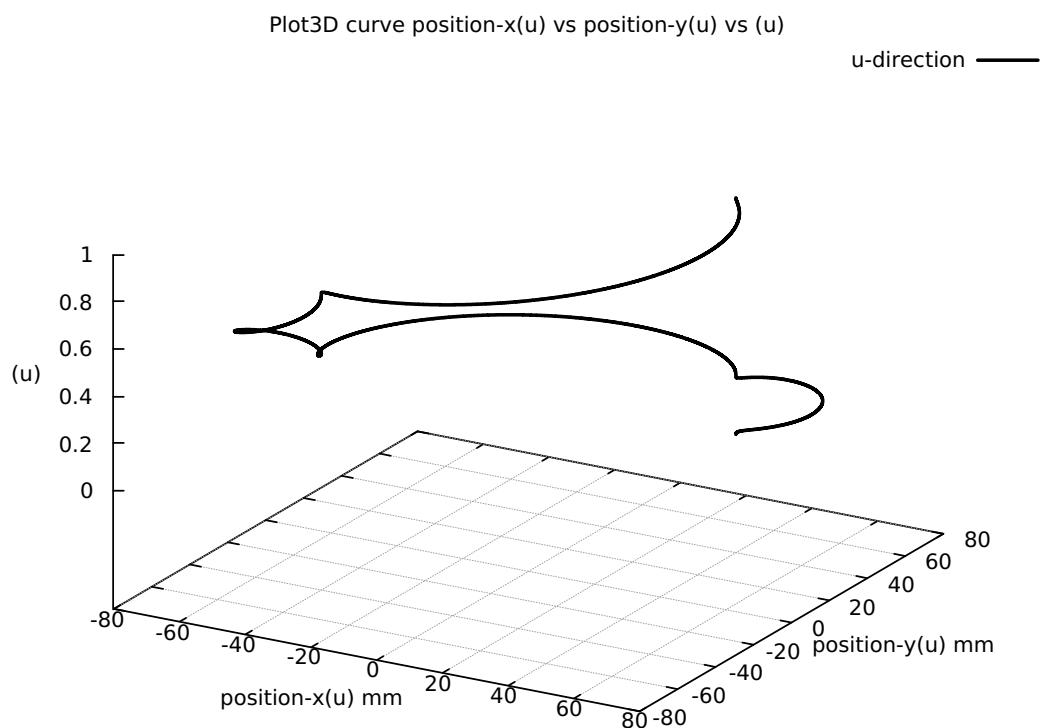


Figure 299: AstEpi First and Second Order Taylor's Approximation

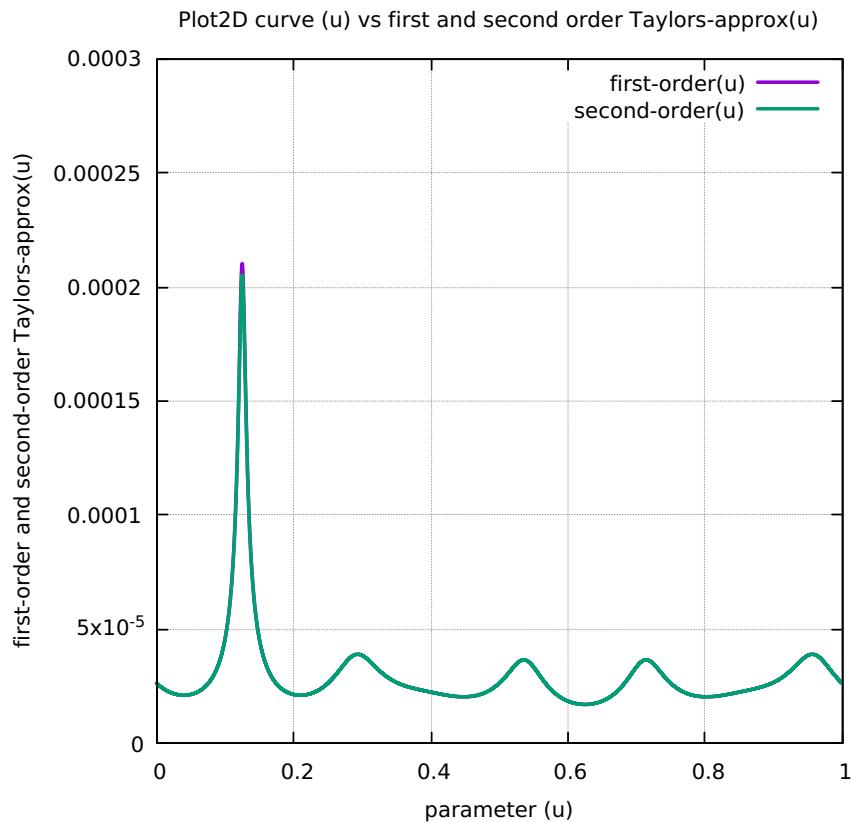


Figure 300: AstEpi First minus Second Order Taylor's Approximation

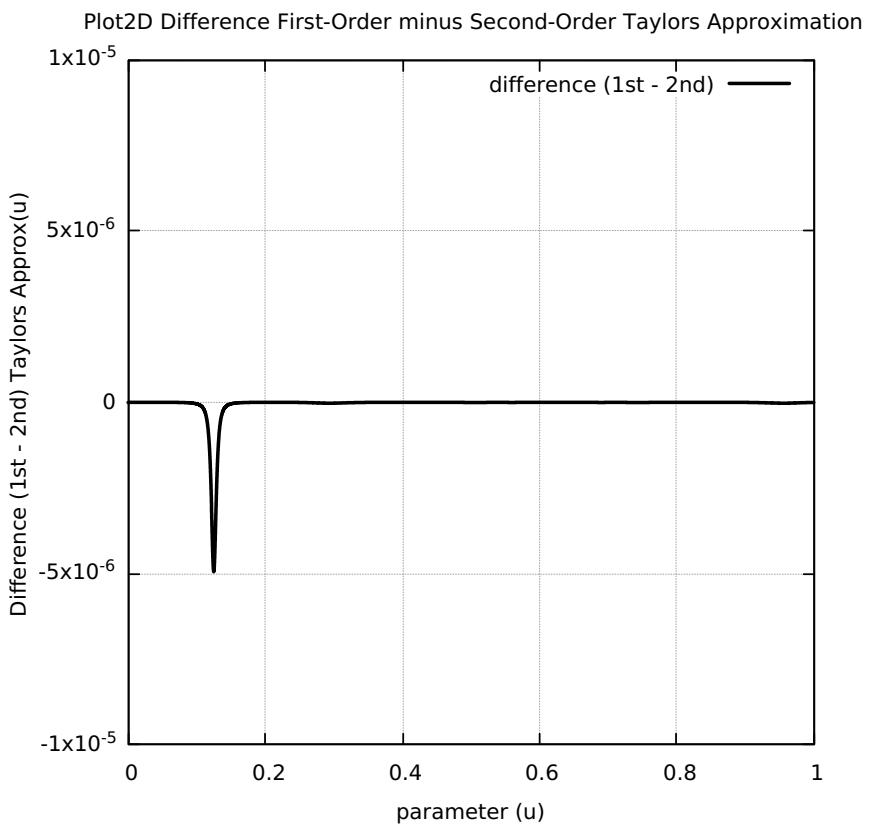


Figure 301: AstEpi Separation First and Second Order Taylor's Approximation

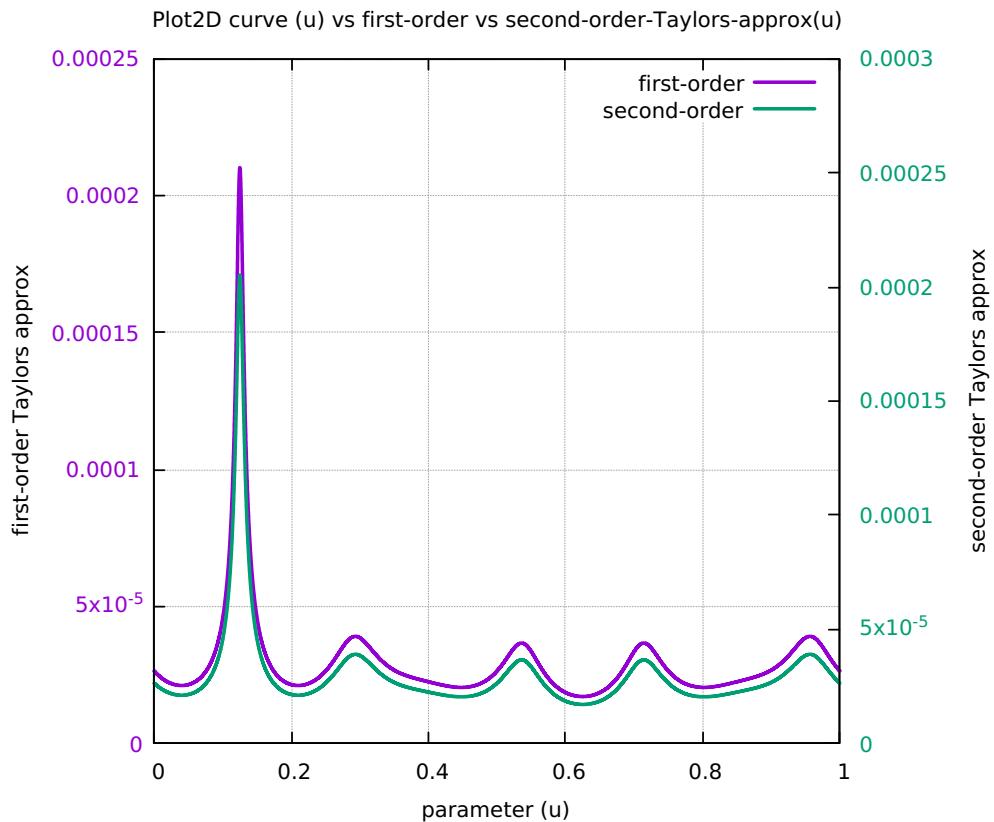


Figure 302: AstEpi Separation SAL and SCL

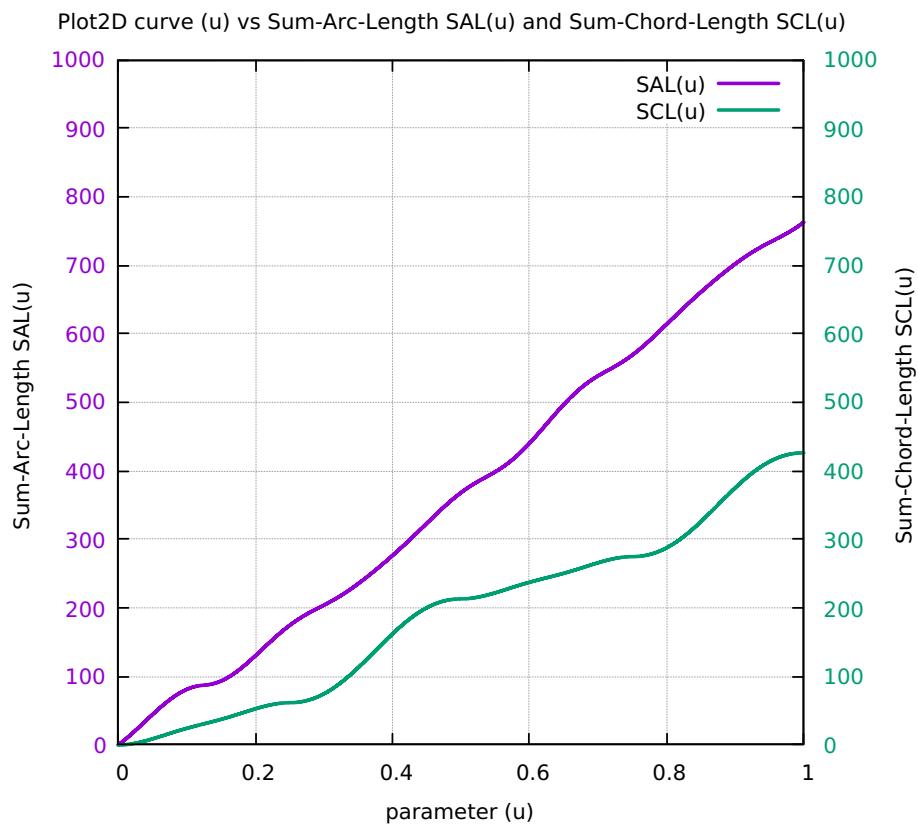


Figure 303: AstEpi Chord-error in close view 2 scales

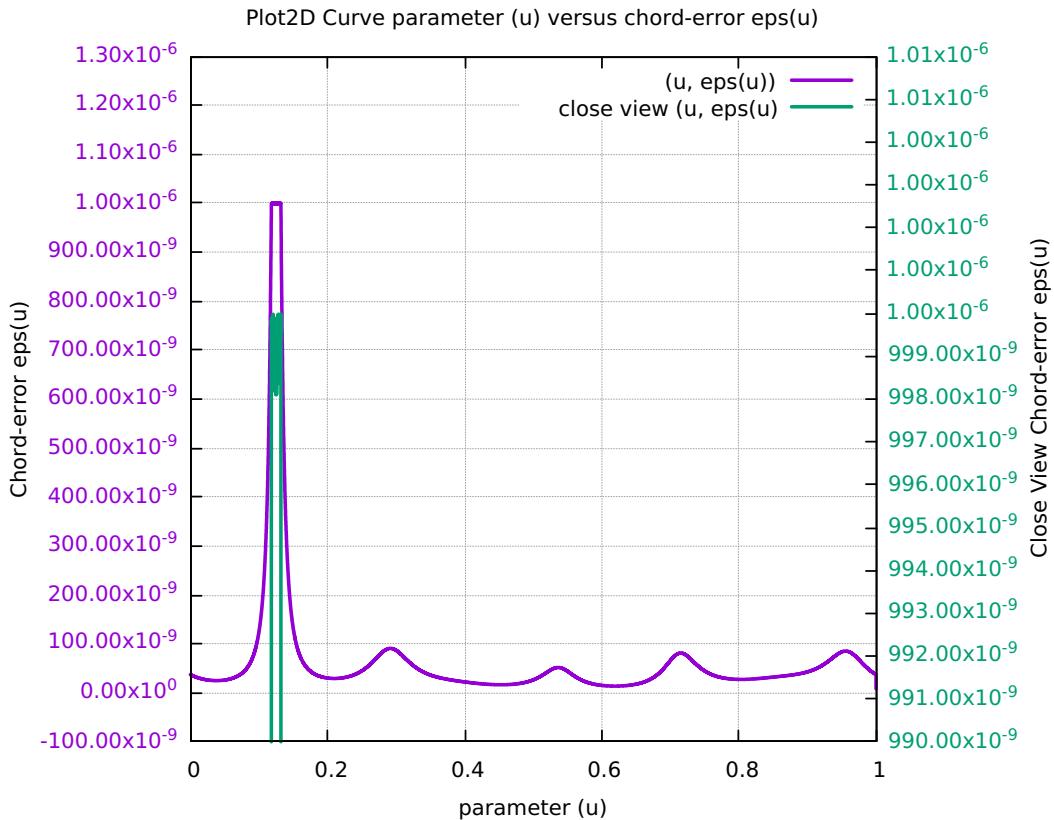


Figure 304: AstEpi Four Components Feedrate Limit

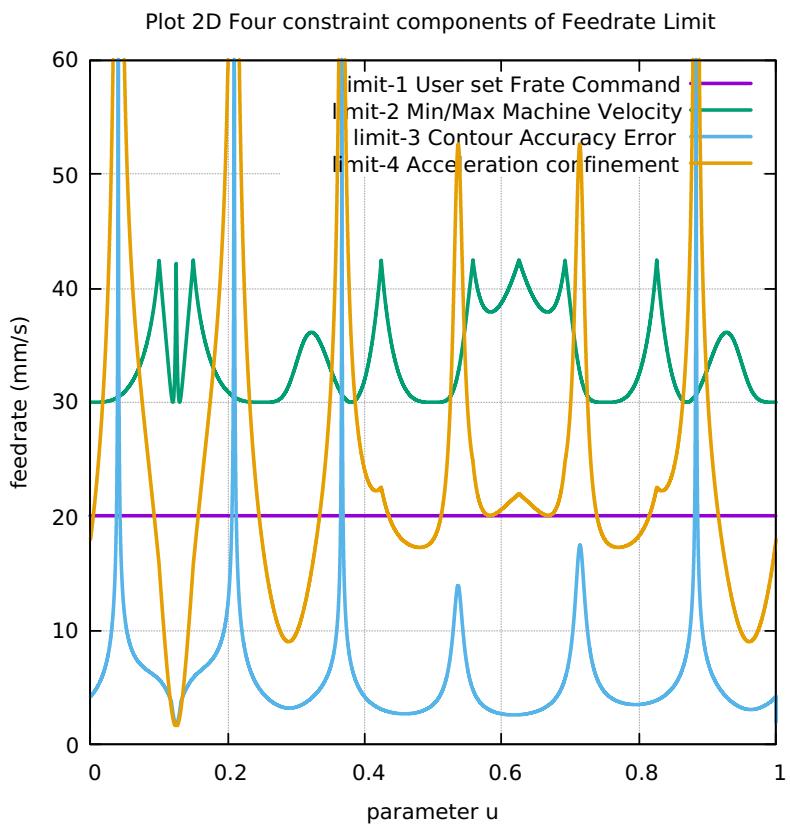


Figure 305: AstEpi FrateCommand FrateLimit and Curr-Frate

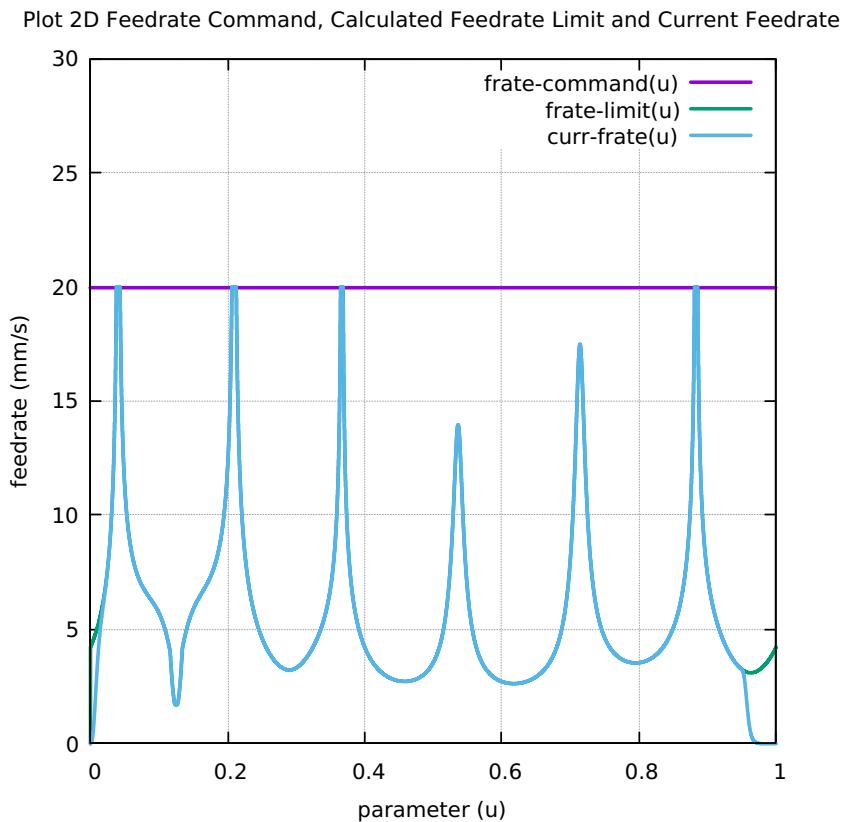


Figure 306: AstEpi FeedRateLimit minus CurrFeedRate

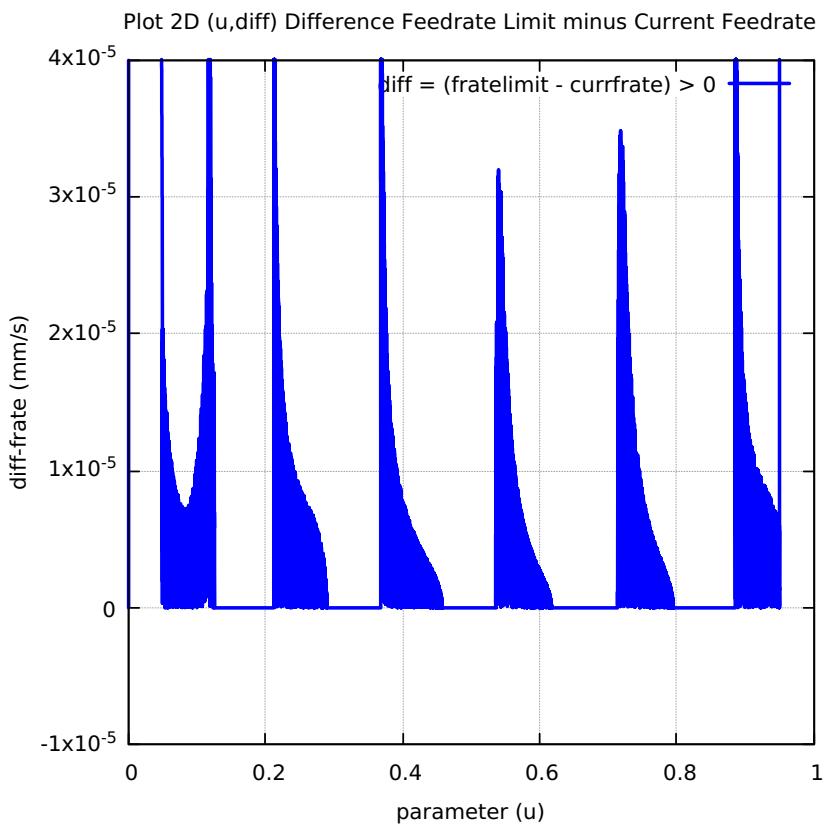


Figure 307: AstEpi FC20-Nominal X and Y Feedrate Profiles

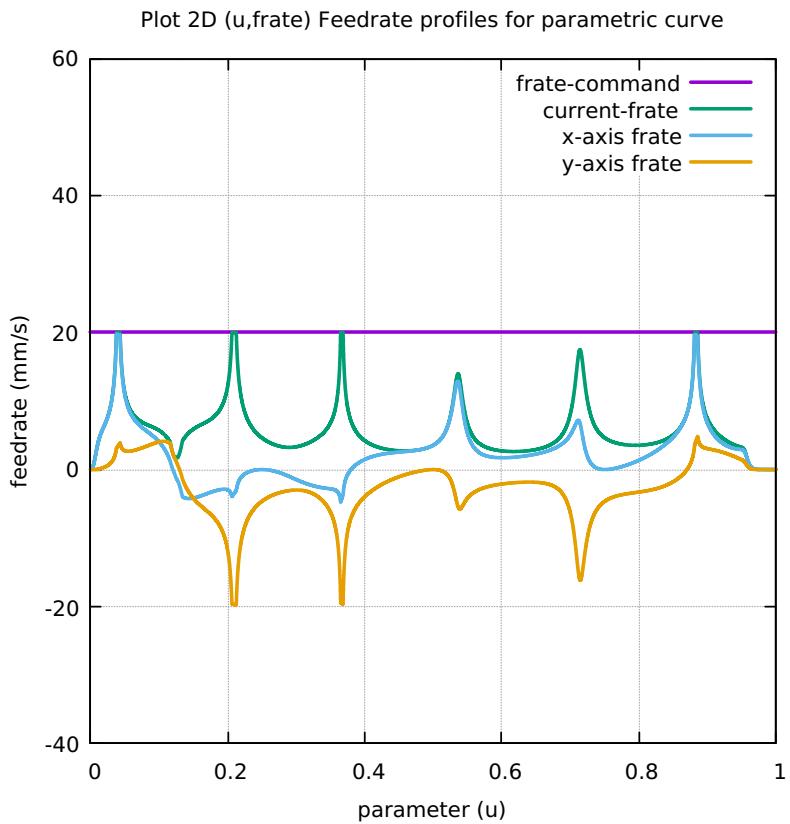


Figure 308: AstEpi FC20 Nominal Tangential Acceleration

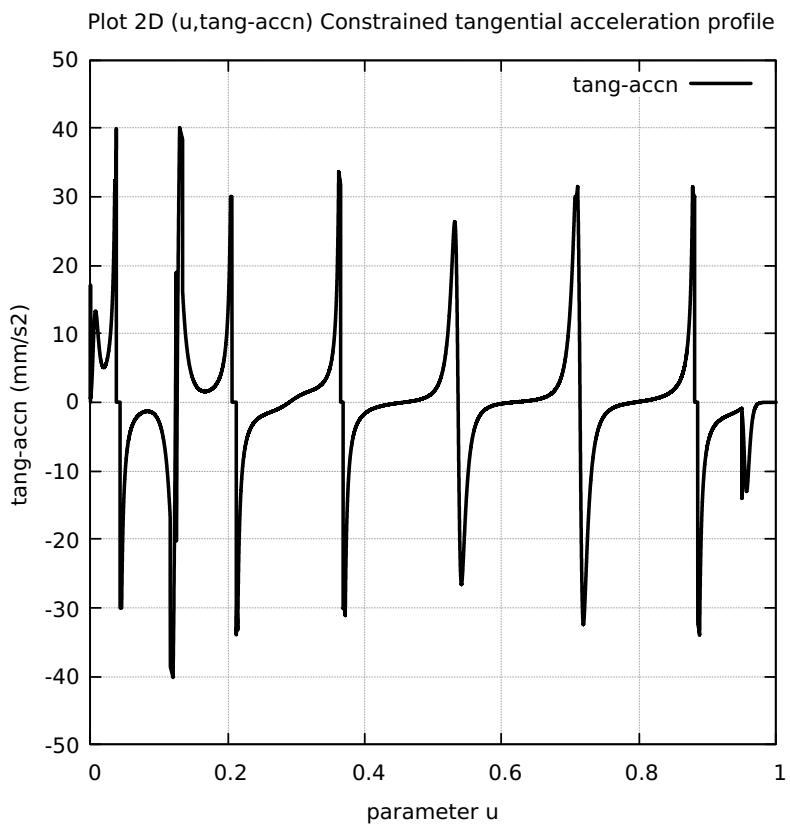


Figure 309: AstEpi FC20 Nominal Rising S-Curve Profile

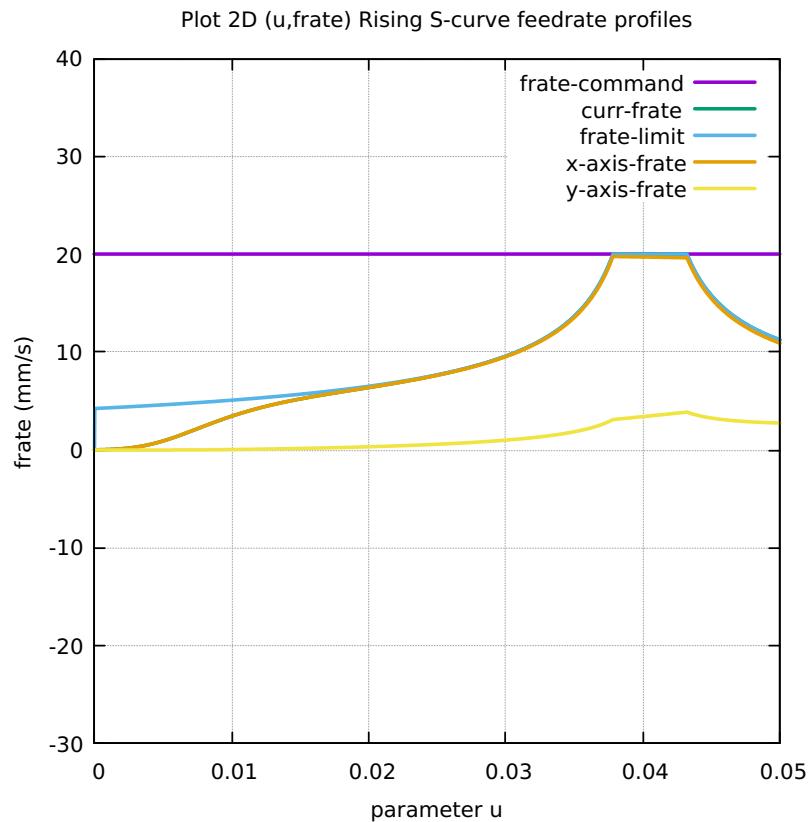


Figure 310: AstEpi FC20 Nominal Falling S-Curve Profile

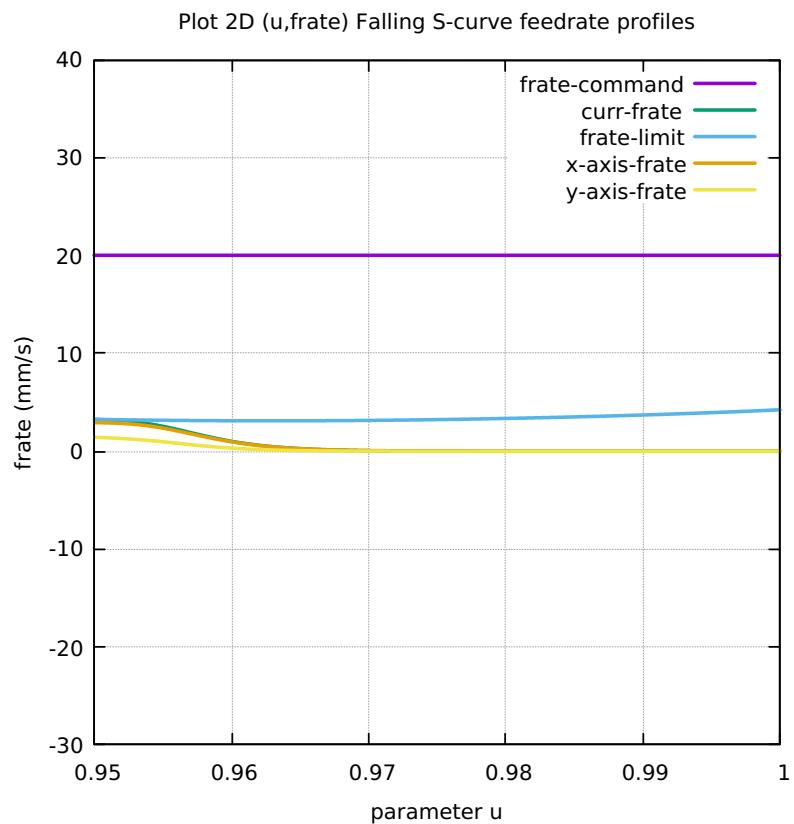


Figure 311: AstEpi FC10 Colored Feedrate Profile data ngcode

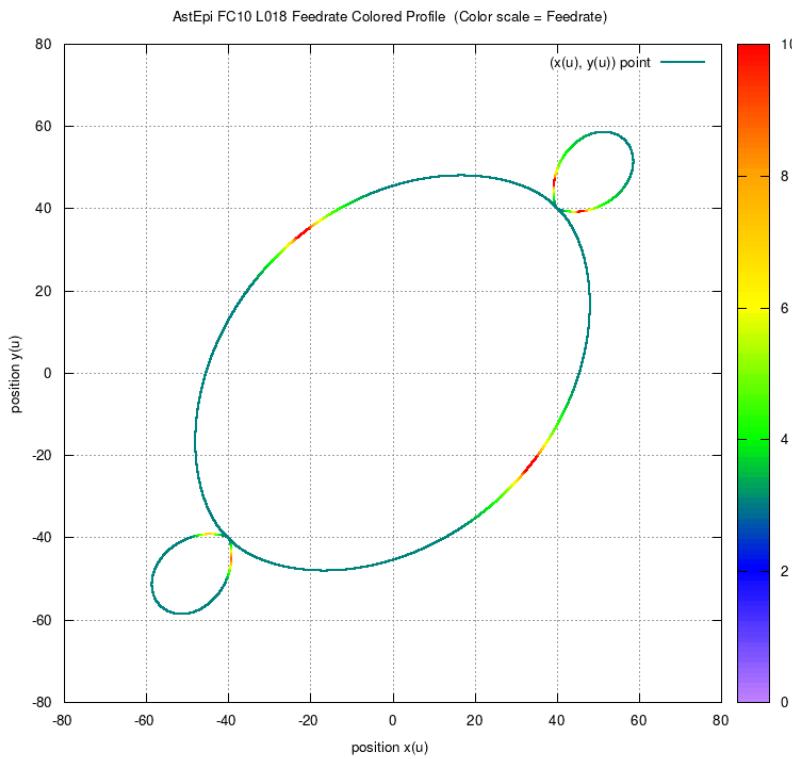


Figure 312: AstEpi FC20 Colored Feedrate Profile data ngcode

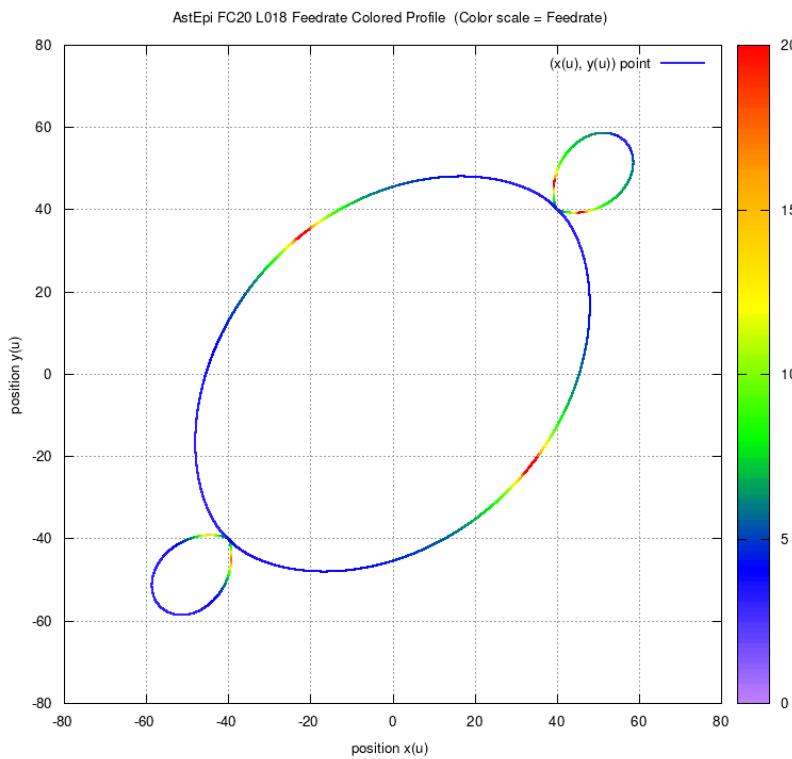


Figure 313: AstEpi FC30 Colored Feedrate Profile data ngcode

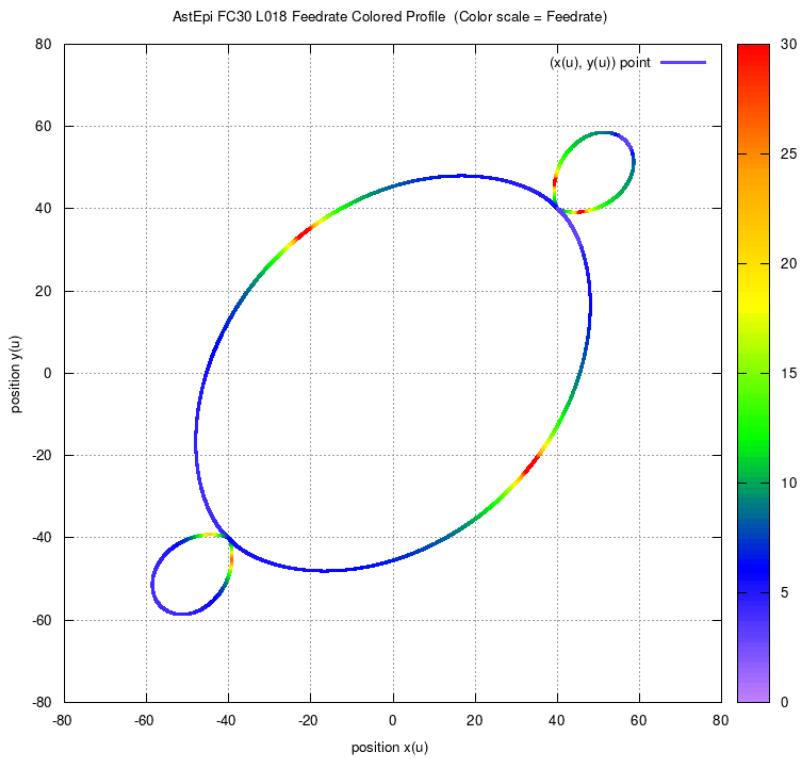


Figure 314: AstEpi FC40 Colored Feedrate Profile data ngcode

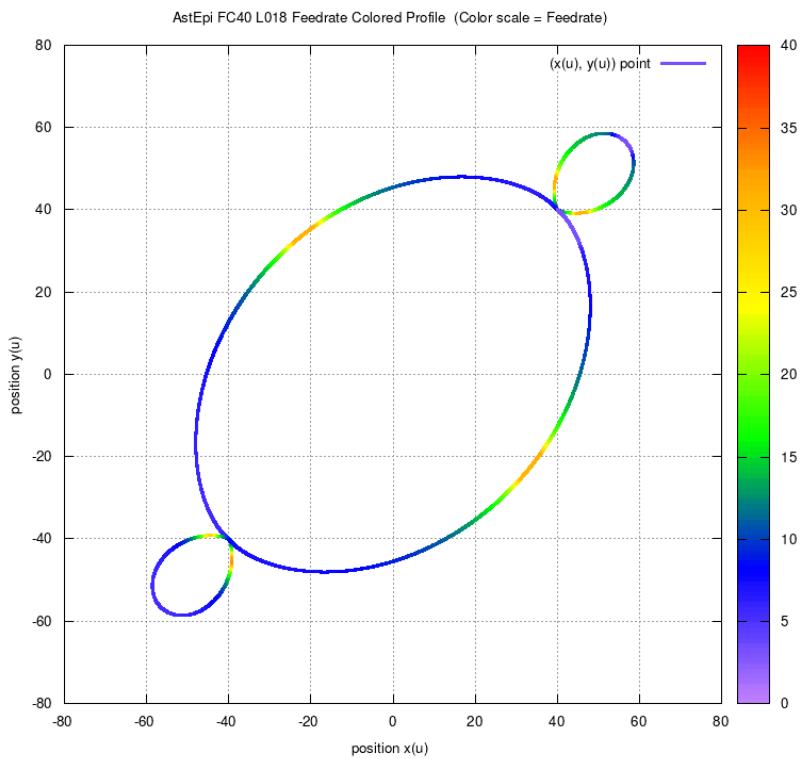


Figure 315: AstEpi FC10 Tangential Acceleration

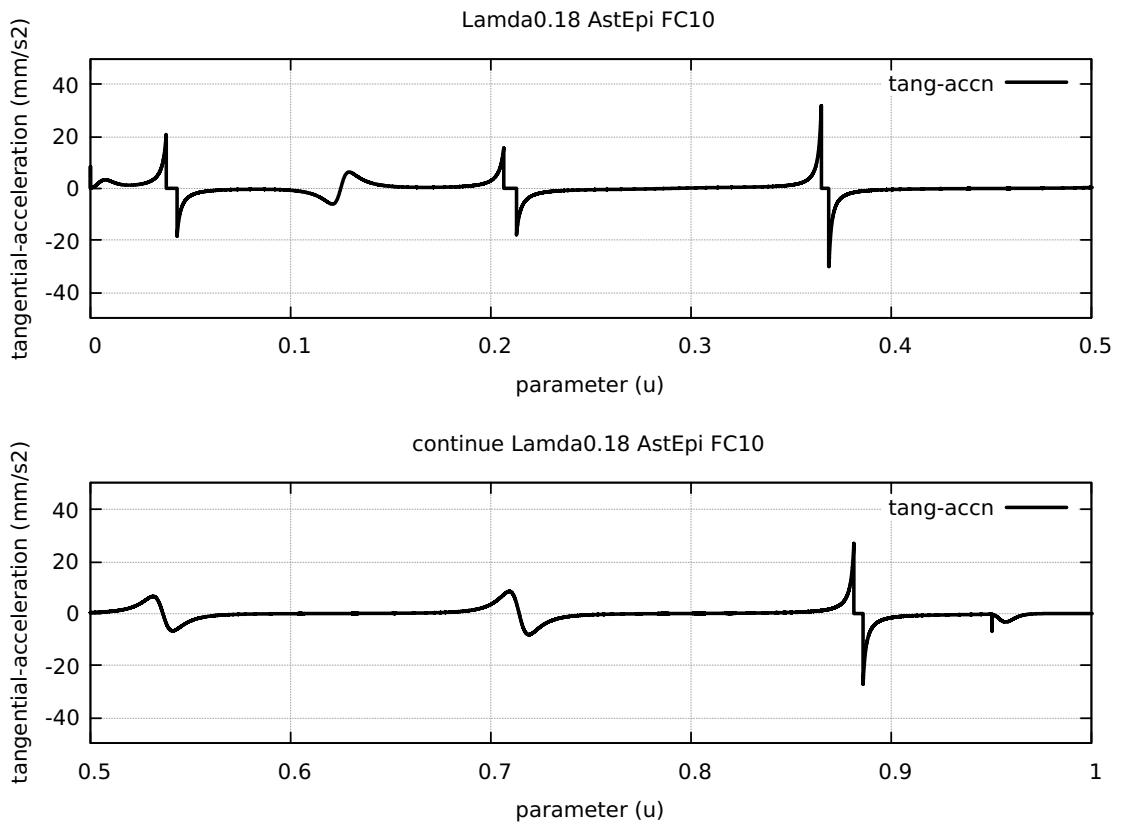


Figure 316: AstEpi FC20 Tangential Acceleration

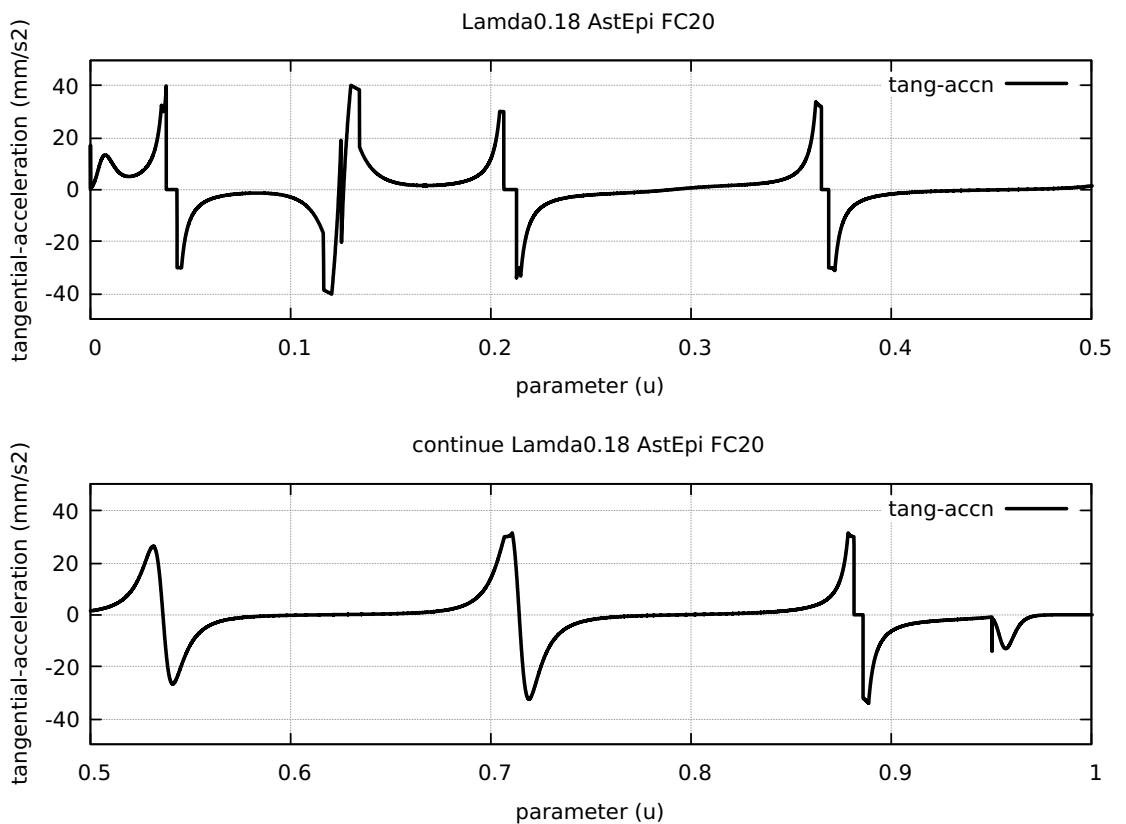


Figure 317: AstEpi FC30 Tangential Acceleration

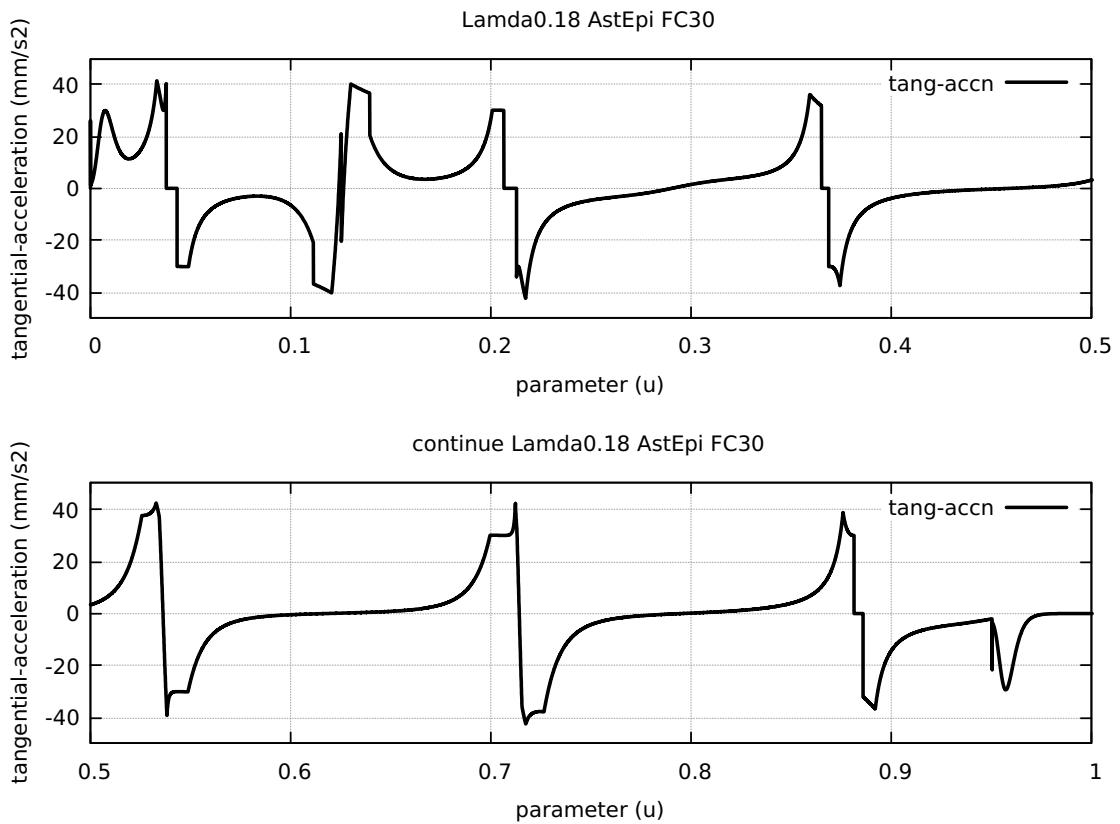


Figure 318: AstEpi FC40 Tangential Acceleration

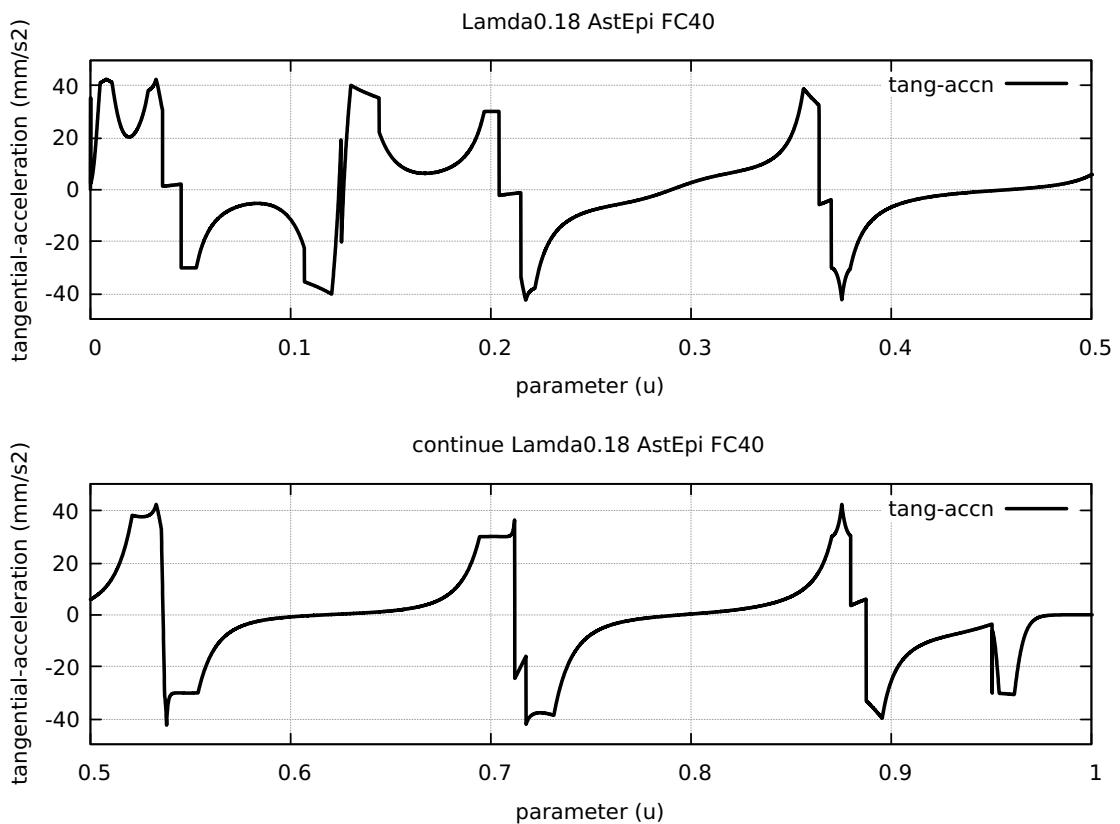


Figure 319: AstEpi FC20 Nominal Separation NAL and NCL

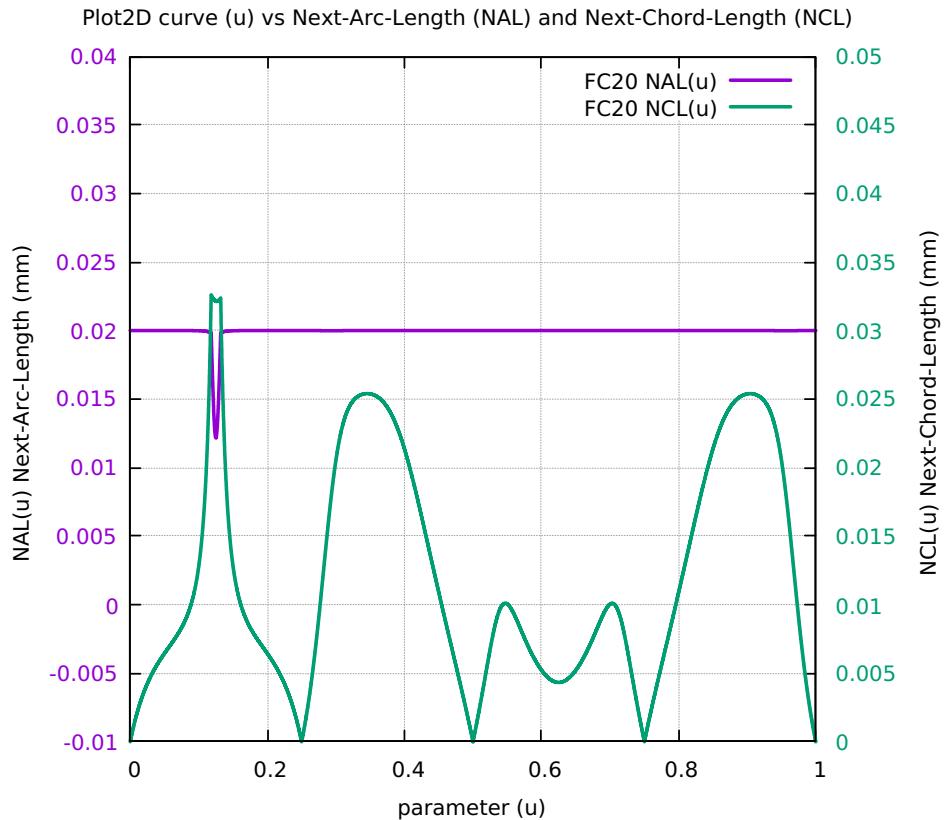


Figure 320: AstEpi Difference SAL minus SCL for FC10 FC20 FC30 FC40

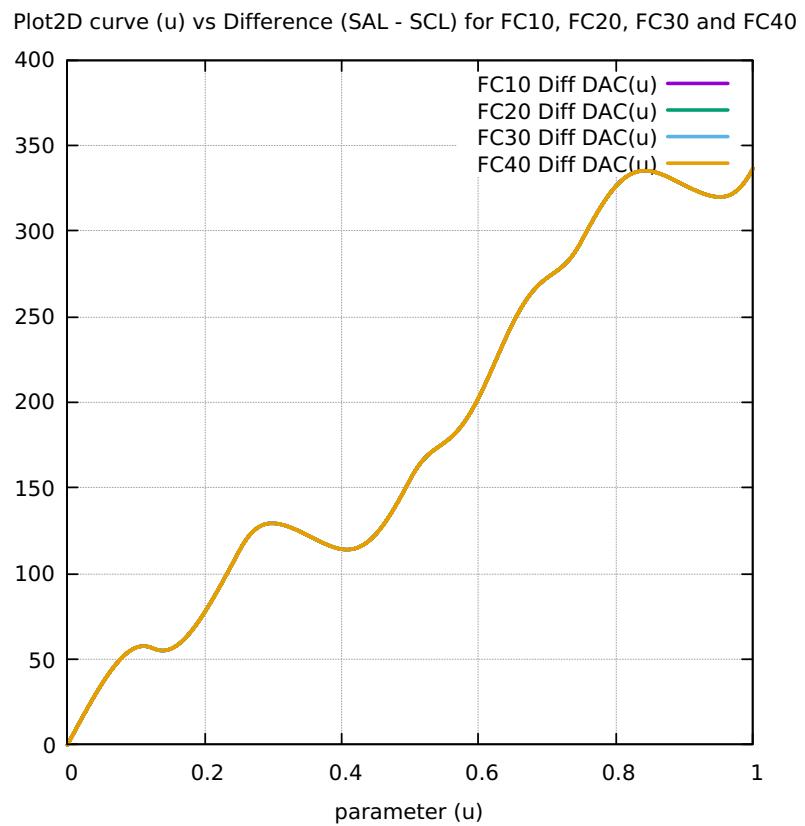


Figure 321: AstEpi FC10 FcateCmd CurrFrate X-Frate Y-Frate

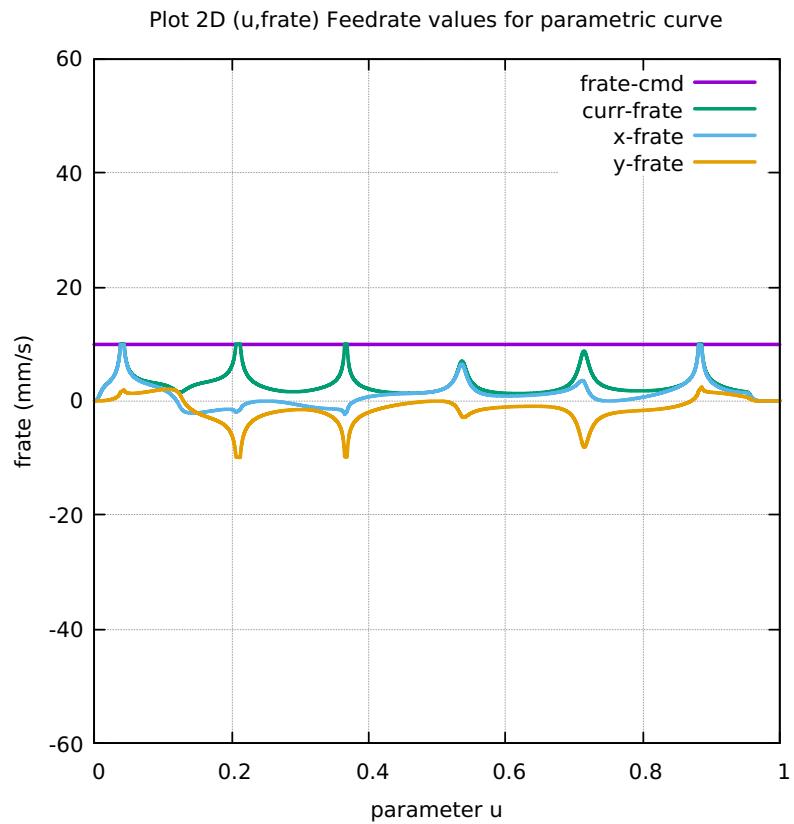


Figure 322: AstEpi FC20 FcateCmd CurrFrate X-Frate Y-Frate

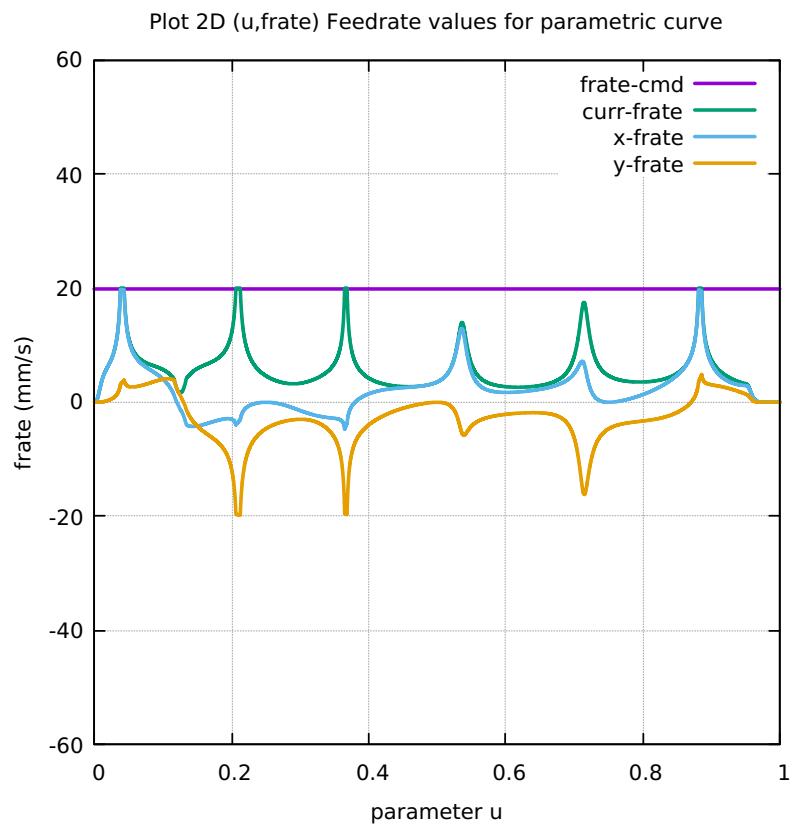


Figure 323: AstEpi FC30 FrateCmd CurrFrate X-Frate Y-Frate

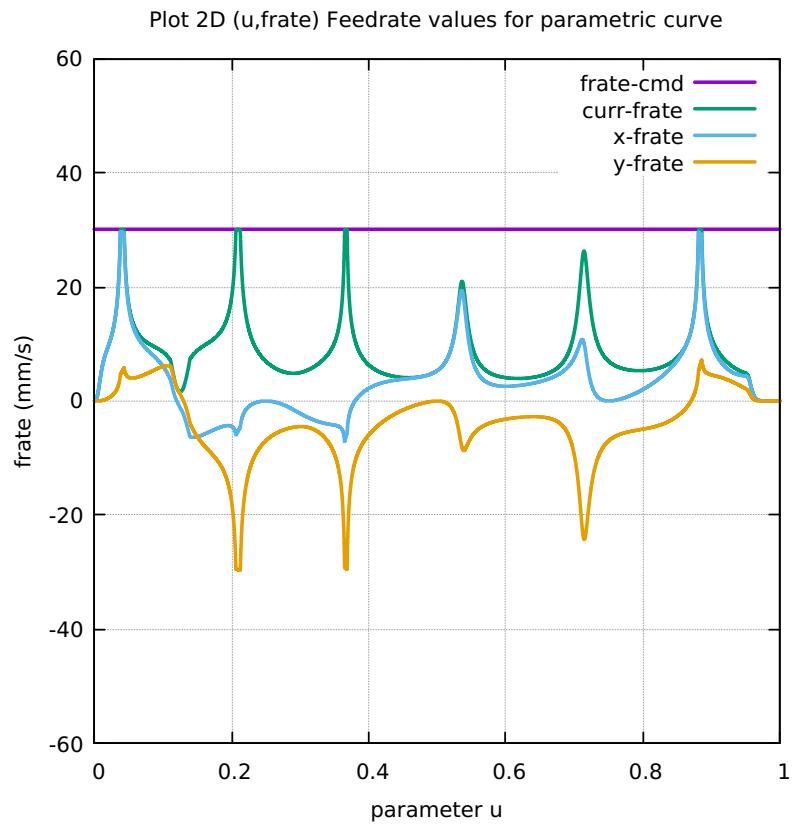


Figure 324: AstEpi FC40 FrateCmd CurrFrate X-Frate Y-Frate

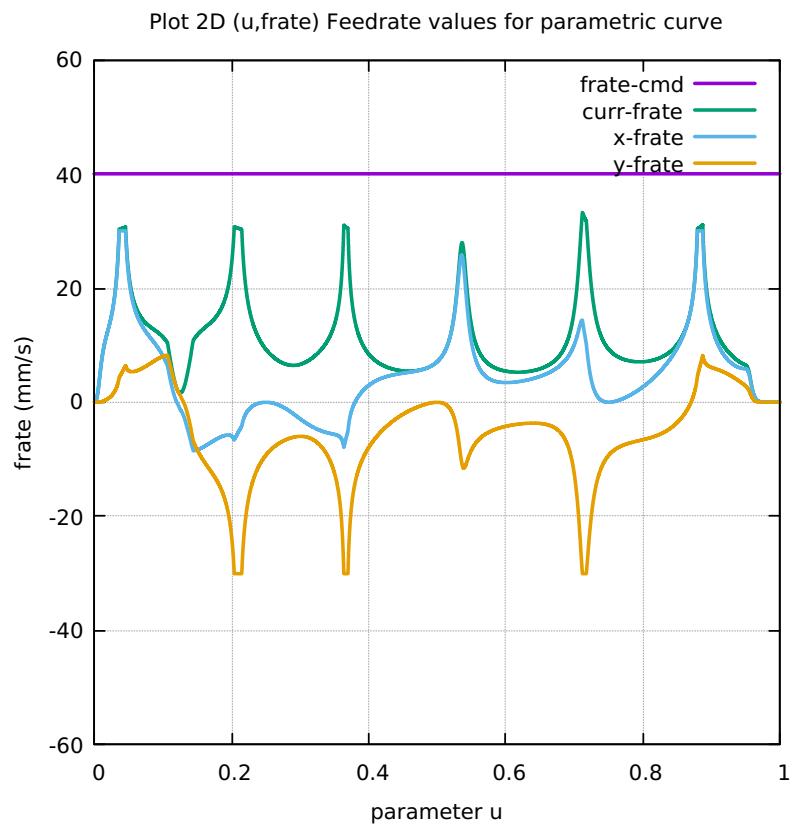


Figure 325: AstEpi FC10 Four Components FeedrateLimit

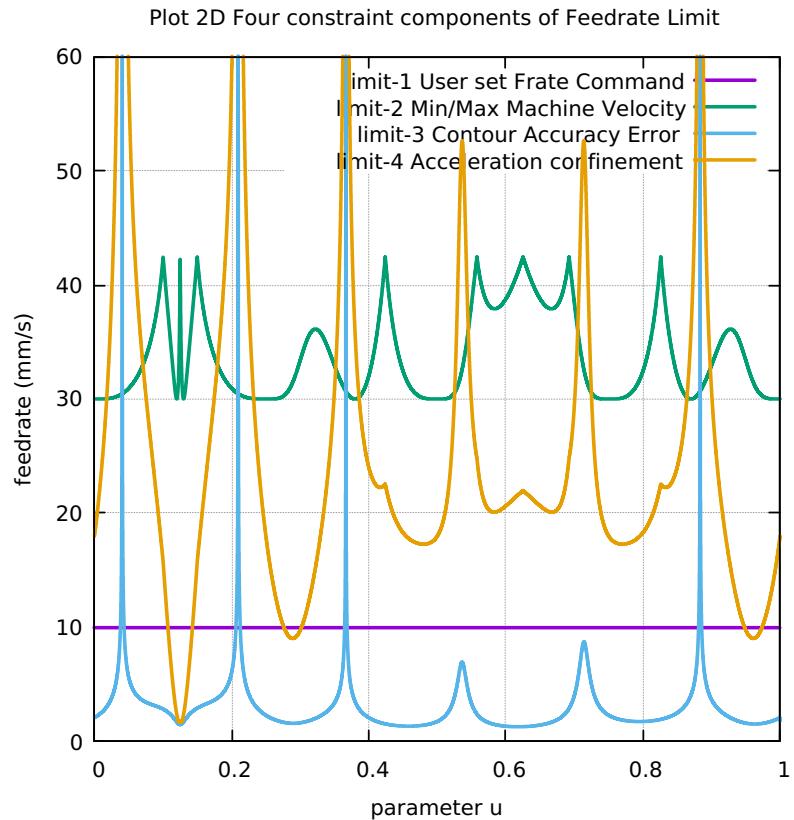


Figure 326: AstEpi FC20 Four Components FeedrateLimit

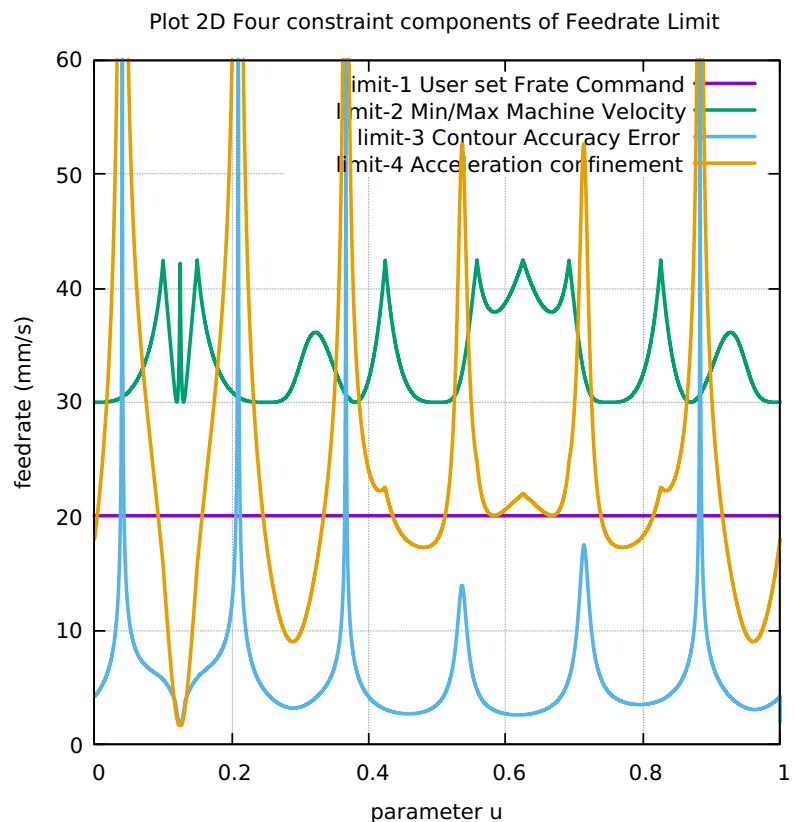


Figure 327: AstEpi FC30 Four Components FeedrateLimit

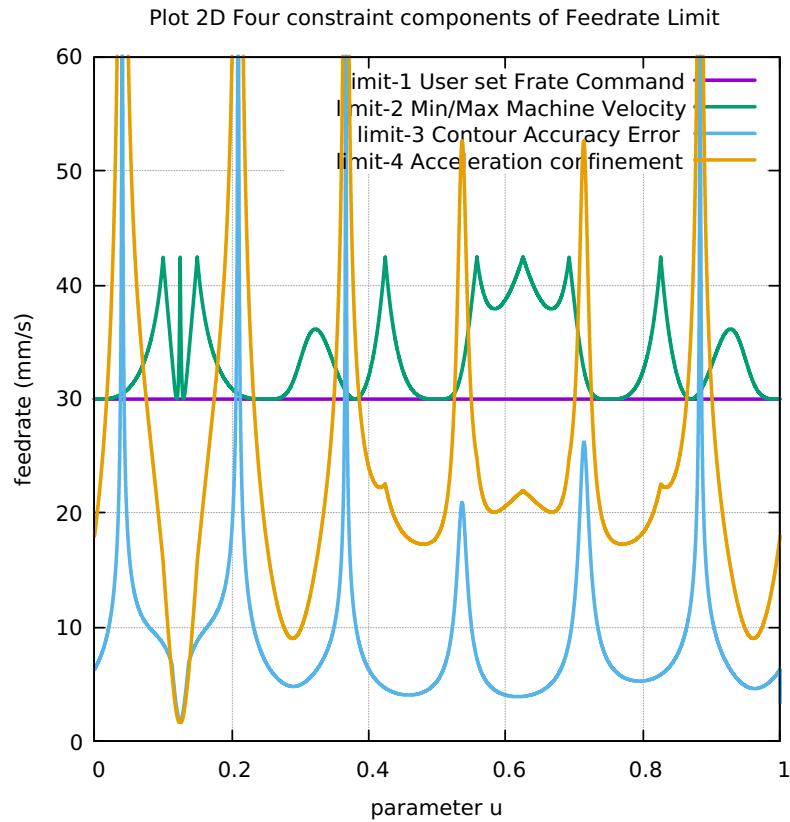


Figure 328: AstEpi FC40 Four Components FeedrateLimit

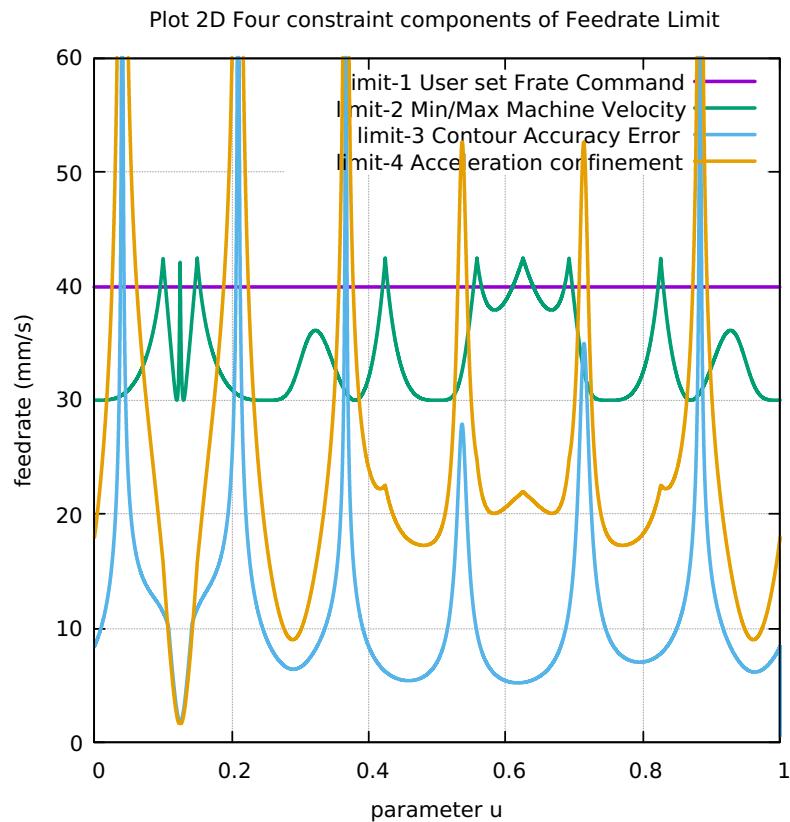


Figure 329: AstEpi Histogram Points FC10 FC20 FC30 FC40

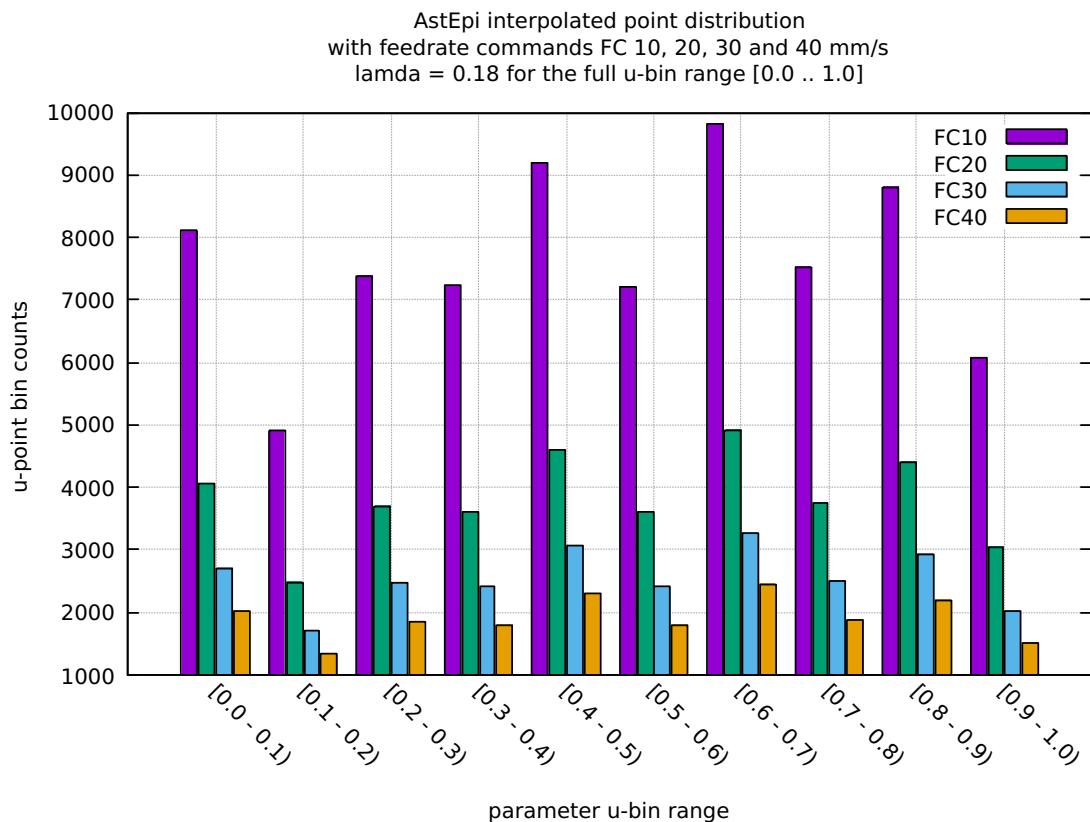


Table 18: AstEpi Table distribution of interpolated points

BINS	FC10	FC20	FC30	FC40
0.0 - 0.1	8110	4055	2704	2028
0.1 - 0.2	4901	2478	1702	1334
0.2 - 0.3	7391	3696	2464	1848
0.3 - 0.4	7234	3617	2412	1809
0.4 - 0.5	9182	4592	3061	2297
0.5 - 0.6	7216	3608	2406	1804
0.6 - 0.7	9831	4916	3278	2458
0.7 - 0.8	7525	3763	2508	1882
0.8 - 0.9	8801	4401	2935	2201
0.9 - 1.0	6084	3043	2029	1523
Tot Counts	76275	38169	25499	19184

Table 19: AstEpi Table FC10-20-30-40 Run Performance data

1	Curve Type	ASTEP1	ASTEP1	ASTEP1
2	User Feedrate Command FC(mm/s)	FC10	FC20	FC40
3	User Lamda Acceleration Safety Factor	0.18	0.18	0.18
4	Total Interpolated Points (TIP)	76275	38169	25499
5	Total Sum-Chord-Error (SCE) (mm)	8.403732211685E-04	1.641842648547E-03	2.390002467065E-03
6	Ratio 1 = (SCE/TIP) = Chord-Error/Point	1.101782024240E-08	4.301620856599E-08	9.373293854673E-08
7	Total Sum-Arc-Length (SAL) (mm)	7.626471894183E+02	7.626564180797E+02	7.626597716718E+02
8	Total Sum-Chord-Length (SCL) (mm)	4.262622478423E+02	4.262622396899E+02	4.262622295515E+02
9	Difference = (SAL - SCL) (mm)	3.363849415760E+02	3.363941783897E+02	3.363975421203E+02
10	Percentage Difference = (SAL - SCL)/SAL	4.410754359858E+01	4.410822100426E+01	4.410846810274E+01
11	Ratio 2 = (SCE/SCL) = Chord Error/Chord-Length	1.971493430212E-06	3.851719659103E-06	5.606883043752E-06
12	Total Sum-Arc-Theta (SAT) (rad)	1.300410711943E+01	1.300415390419E+01	1.300401093322E+01
13	Total Sum-Arc-Area (SAA) (mm2)	2.898515736808E-04	6.811935608835E-04	9.367445601128E-04
14	Ratio 3 = (SAA/SCL) = Arc-Area/Chord-Length	1.971493430212E-06	3.851719659103E-06	5.606883043752E-06
15	Average-Chord-Error (ACE) (mm)	1.101782024240E-08	4.301620856599E-08	9.373293854673E-08
16	Average-Arc-Length (AAL) (mm)	9.998783195037E-03	1.998156618318E-02	2.991057226731E-02
17	Average-Chord-Length (ACL) (mm)	5.588565537959E-03	1.116805281099E-02	1.671747703944E-02
18	Average-Arc-Theta (AAT) (rad)	1.704920040831E-04	3.407082871566E-04	5.100012131627E-04
19	Average-Arc-Area (AAA) (mm2)	3.800136005465E-09	1.784724273956E-08	3.673796219754E-08
20	Algorithm actual runtime on computer (ART) (s)	27.836373672	14.372286925	9.138001576
				7.71877336

.12 APPENDIX SNAHYP CURVE

- .12.1 Plot of SnaHyp curve [330]
- .12.2 SnaHyp Radius of Curvature [331]
- .12.3 SnaHyp Validation in LinuxCNC [332]
- .12.4 SnaHyp Direction of Travel 3D [333]
- .12.5 SnaHyp First and Second Order Taylors App [334]
- .12.6 SnaHyp First minus Second Order Taylors App [335]
- .12.7 SnaHyp Separate First Second Order Taylors App [336]
- .12.8 SnaHyp Separation SAL and SCL [337]
- .12.9 SnaHyp Chord-error in close view 2 scales [338]
- .12.10 SnaHyp Four Components Feedrate Limit [339]
- .12.11 SnaHyp FrateCommand FrateLimit and Curr-Frate [340]
- .12.12 SnaHyp FeedRateLimit minus CurrFeedRate [341]
- .12.13 SnaHyp FC20-Nominal X and Y Feedrate Profiles [342]
- .12.14 SnaHyp FC20 Nominal Tangential Acceleration [343]
- .12.15 SnaHyp FC20 Nominal Rising S-Curve Profile [344]
- .12.16 SnaHyp FC20 Nominal Falling S-Curve Profile [345]
- .12.17 SnaHyp FC10 Colored Feedrate Profile ngcode [346]
- .12.18 SnaHyp FC20 Colored Feedrate Profile ngcode [347]

- .12.19 SnaHyp FC25 Colored Feedrate Profile ngcode [348]
- .12.20 SnaHyp FC28 Colored Feedrate Profile ngcode [349]
- .12.21 SnaHyp FC10 Tangential Acceleration [350]
- .12.22 SnaHyp FC20 Tangential Acceleration [351]
- .12.23 SnaHyp FC30 Tangential Acceleration [352]
- .12.24 SnaHyp FC40 Tangential Acceleration [353]
- .12.25 SnaHyp FC20 Nominal Separation NAL and NCL [354]
- .12.26 SnaHyp SAL minus SCL for FC10 FC20 FC30 FC40 [355]
- .12.27 SnaHyp FC10 FrateCmd CurrFrate X-Frate Y-Frate [356]
- .12.28 SnaHyp FC20 FrateCmd CurrFrate X-Frate Y-Frate [357]
- .12.29 SnaHyp FC25 FrateCmd CurrFrate X-Frate Y-Frate [358]
- .12.30 SnaHyp FC28 FrateCmd CurrFrate X-Frate Y-Frate [359]
- .12.31 SnaHyp FC10 Four Components FeedrateLimit [360]
- .12.32 SnaHyp FC20 Four Components FeedrateLimit [361]
- .12.33 SnaHyp FC25 Four Components FeedrateLimit [362]
- .12.34 SnaHyp FC28 Four Components FeedrateLimit [363]
- .12.35 SnaHyp Histogram Points FC10 FC20 FC30 FC40 [364]
- .12.36 SnaHyp Table distribution of interpolated points [20]
- .12.37 SnaHyp Table FC10-20-30-40 Run Performance data [21]

Figure 330: Plot of SnaHyp curve

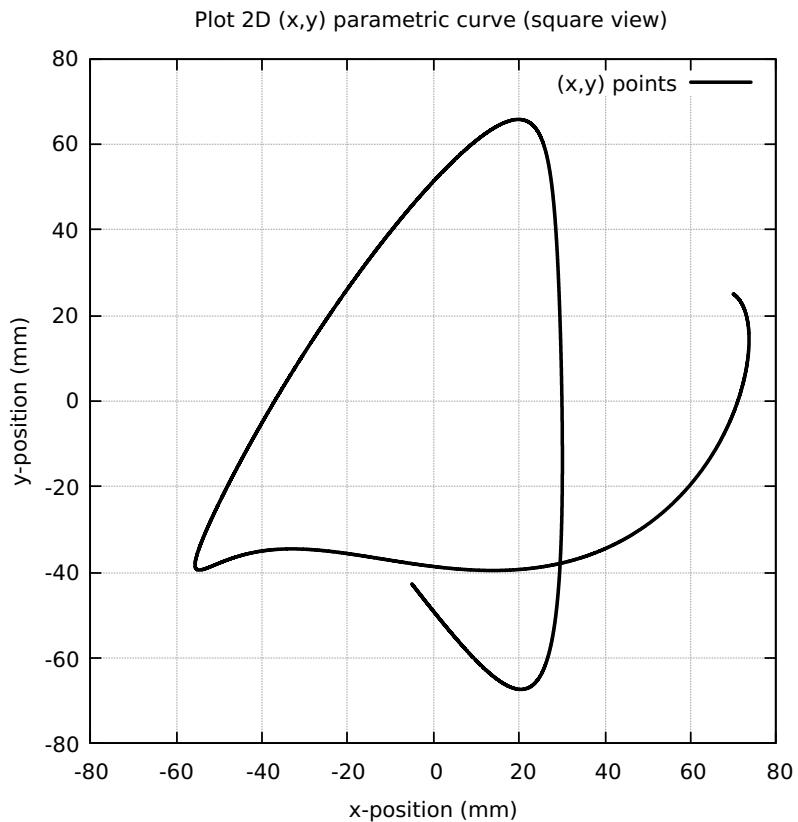


Figure 331: SnaHyp Radius of Curvature

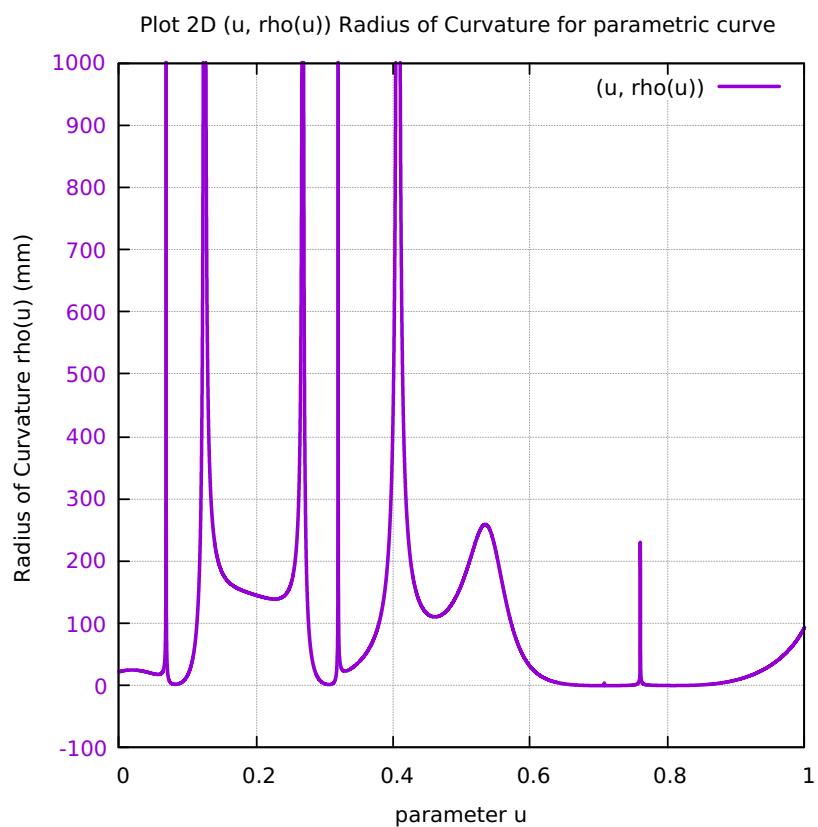


Figure 332: SnaHyp Validation in LinuxCNC

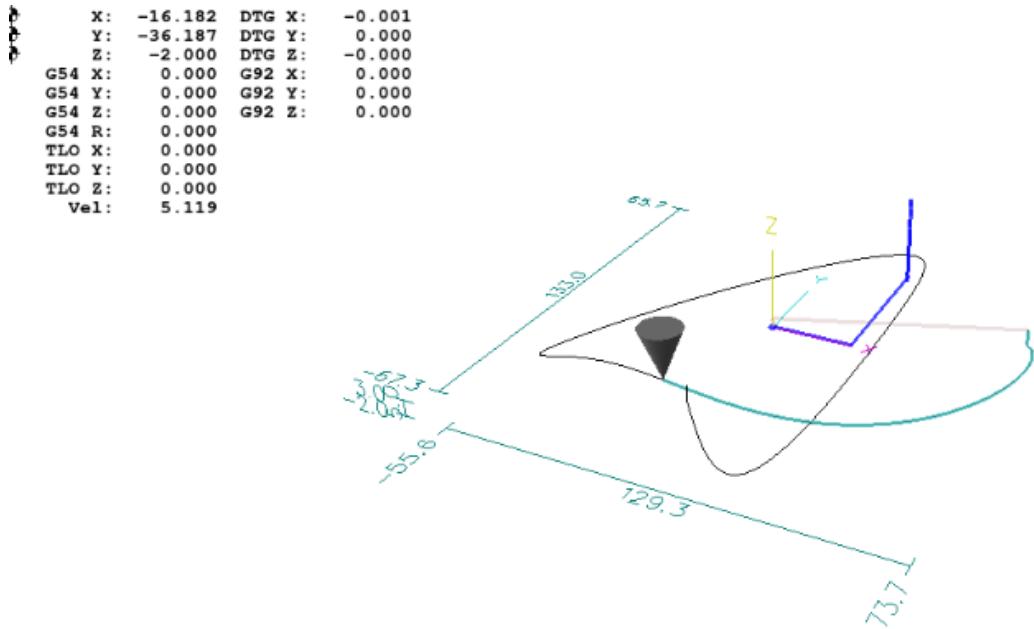


Figure 333: SnaHyp Direction of Travel 3D

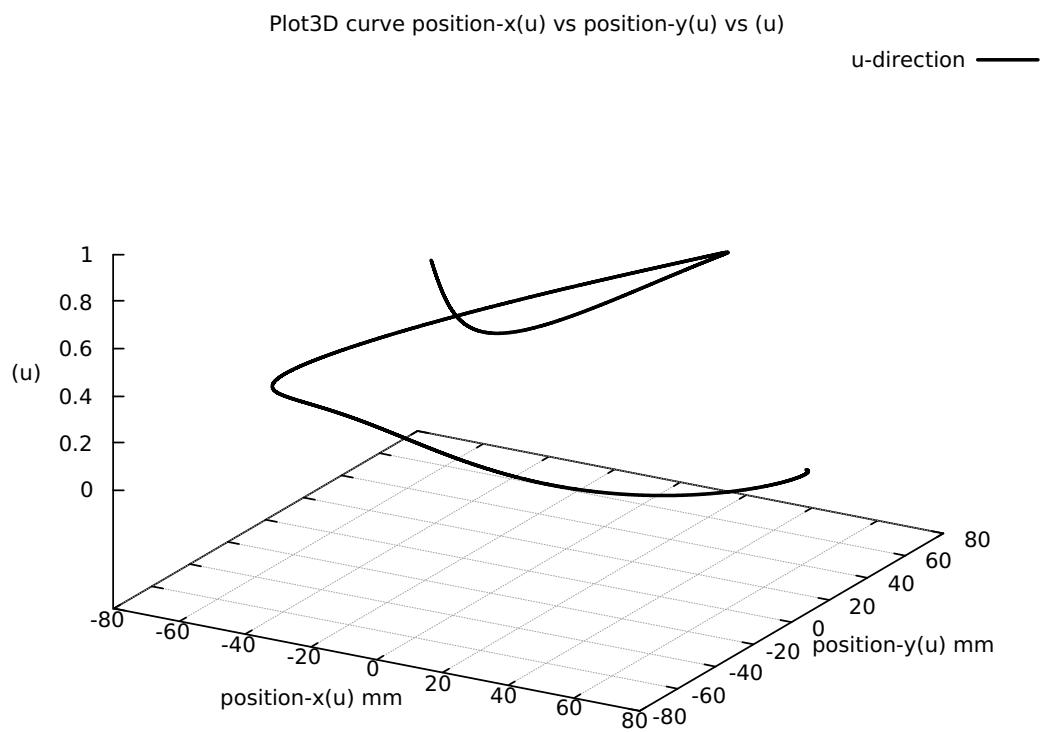


Figure 334: SnaHyp First and Second Order Taylor's Approximation

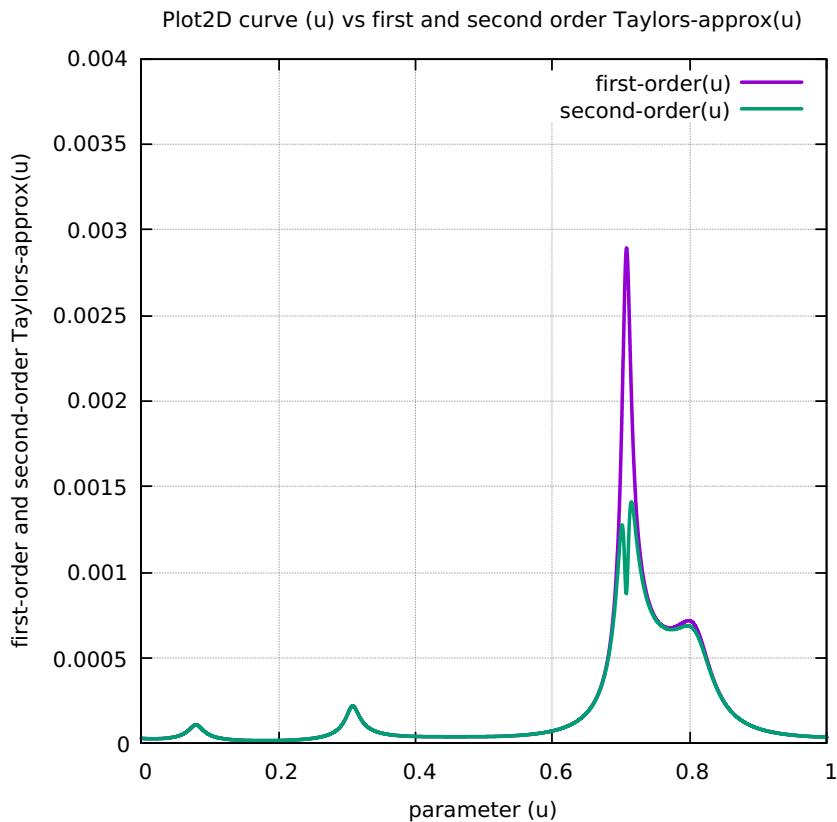


Figure 335: SnaHyp First minus Second Order Taylor's Approximation

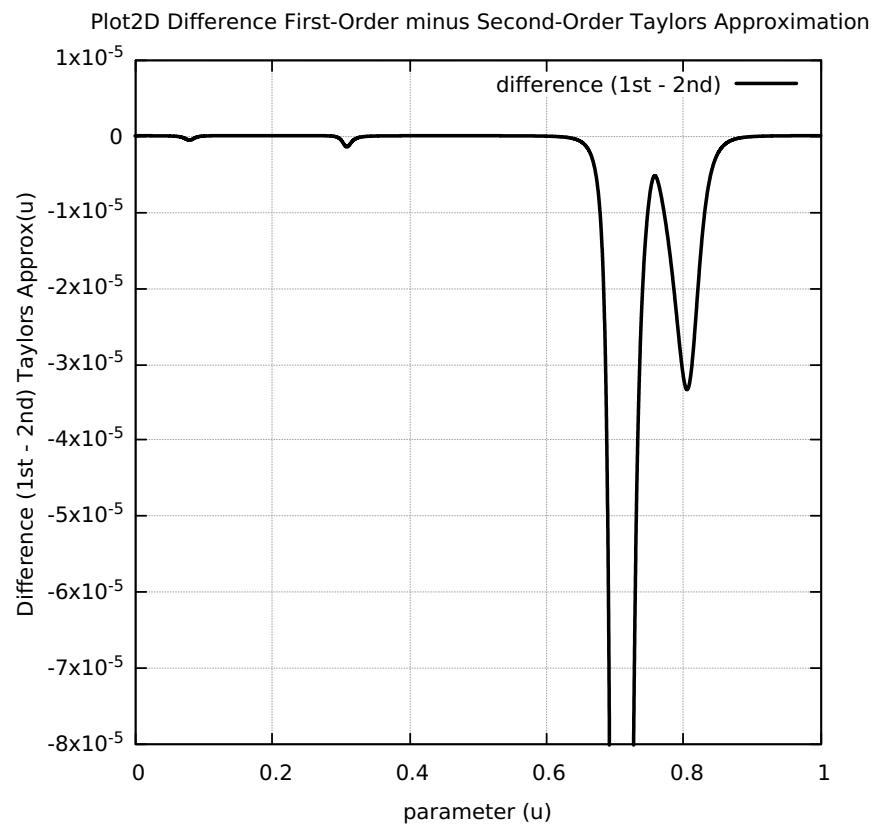


Figure 336: SnaHyp Separation First and Second Order Taylor's Approximation

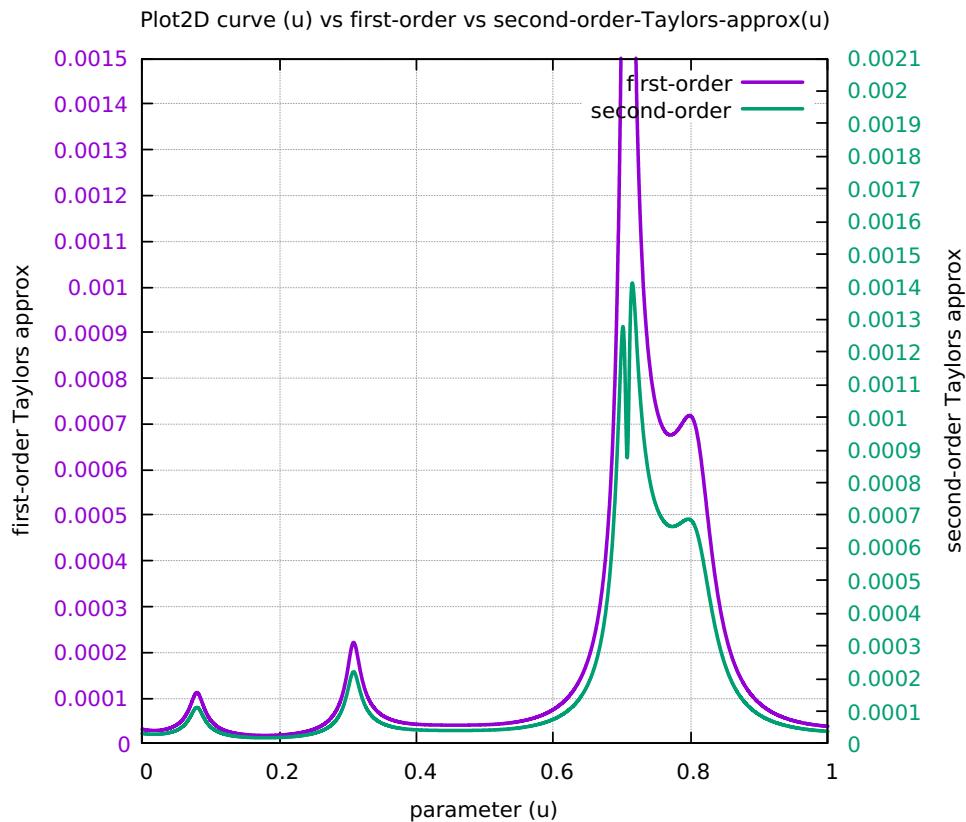


Figure 337: SnaHyp Separation SAL and SCL

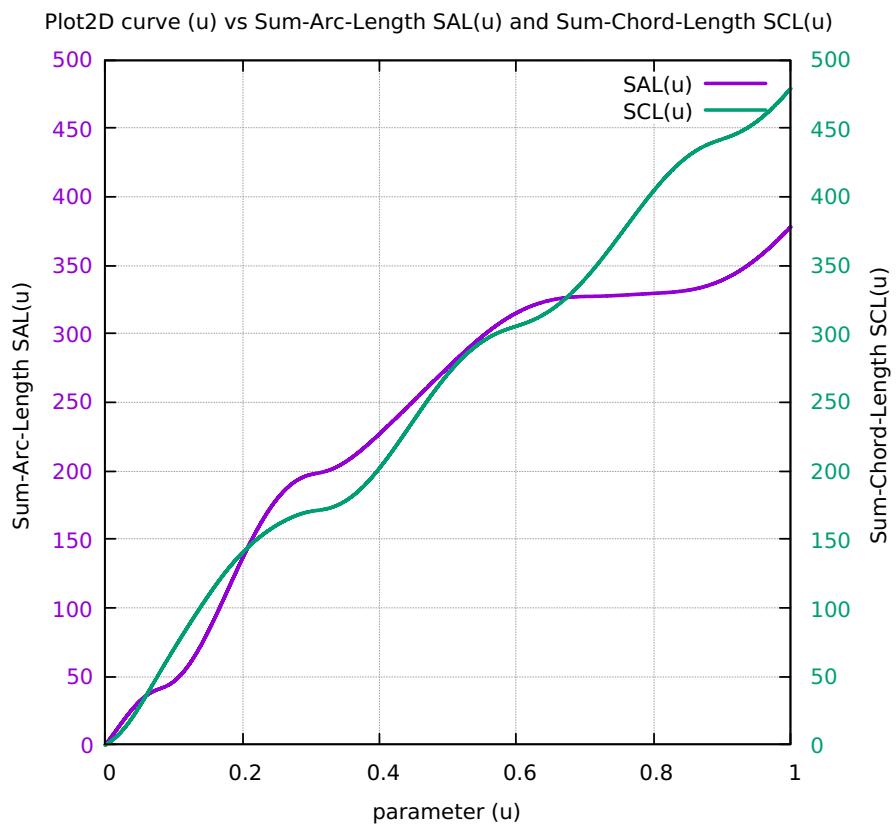


Figure 338: SnaHyp Chord-error in close view 2 scales

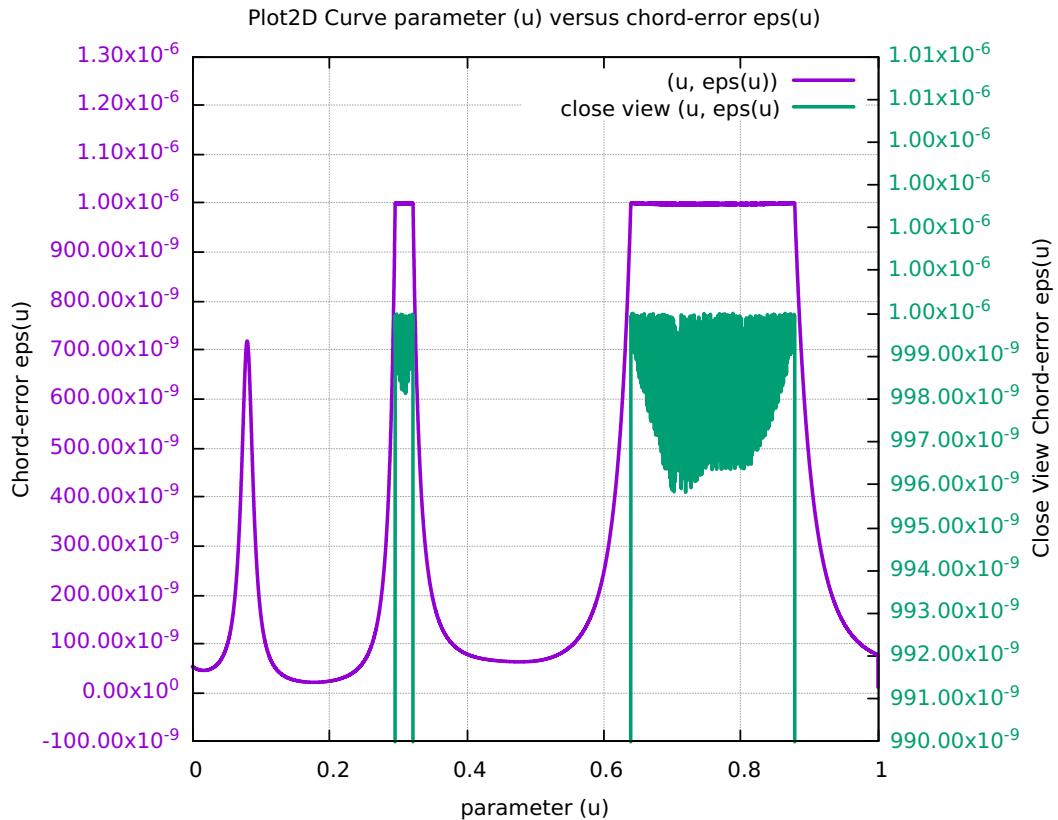


Figure 339: SnaHyp Four Components Feedrate Limit

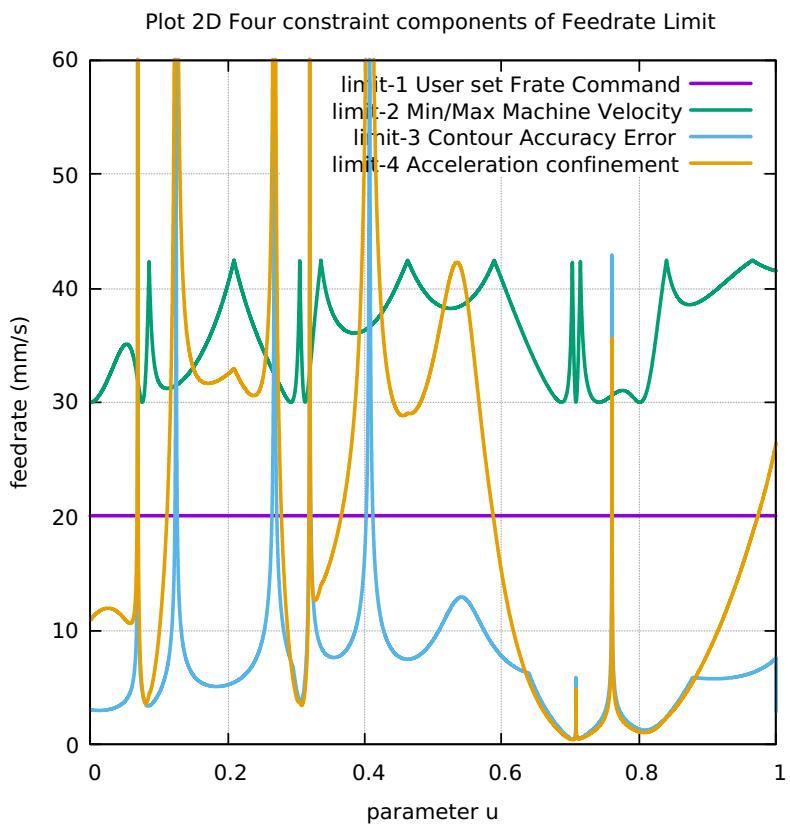


Figure 340: SnaHyp FrateCommand FrateLimit and Curr-Frate

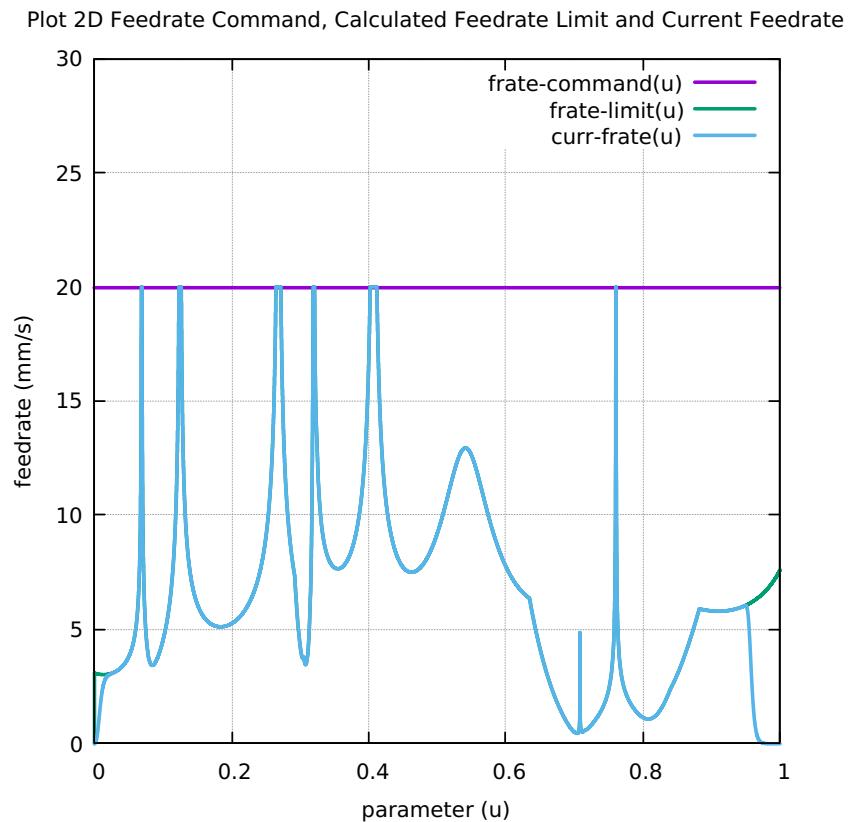


Figure 341: SnaHyp FeedRateLimit minus CurrFeedRate

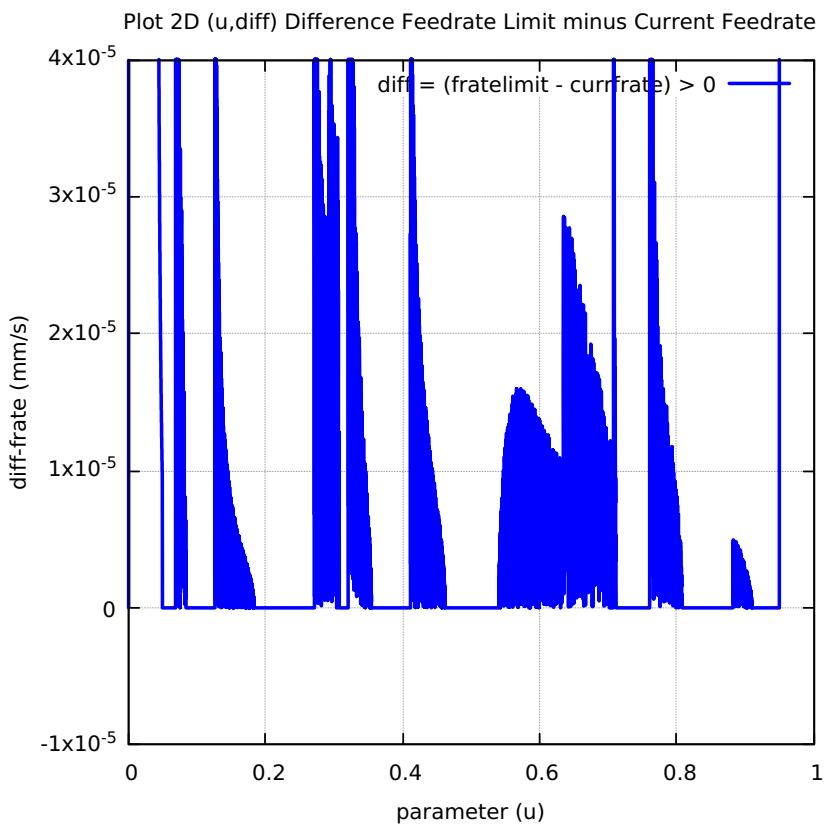


Figure 342: SnaHyp FC20-Nominal X and Y Feedrate Profiles

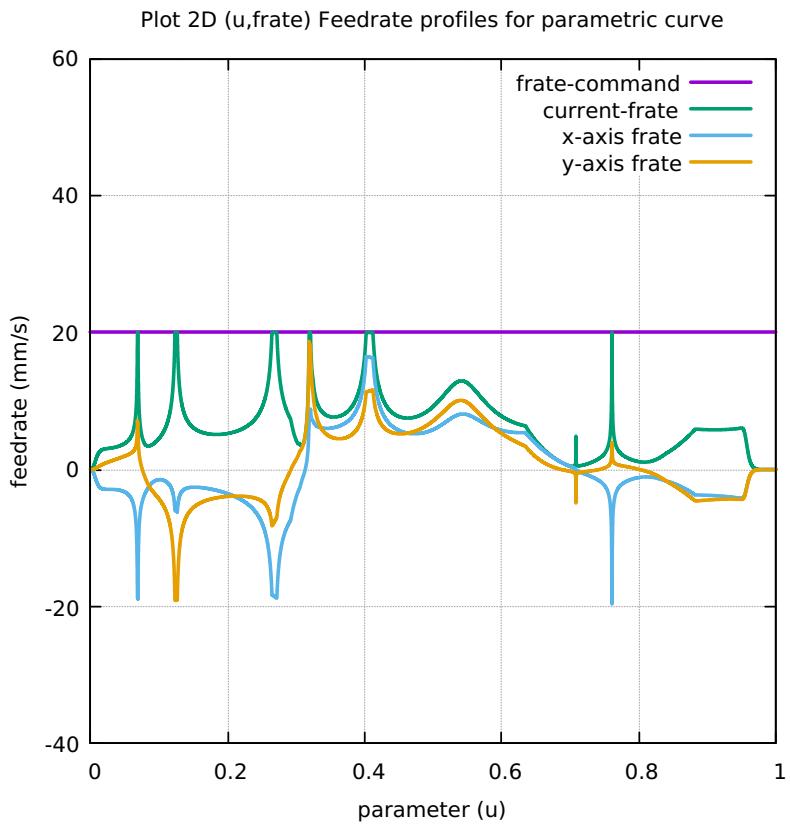


Figure 343: SnaHyp FC20 Nominal Tangential Acceleration

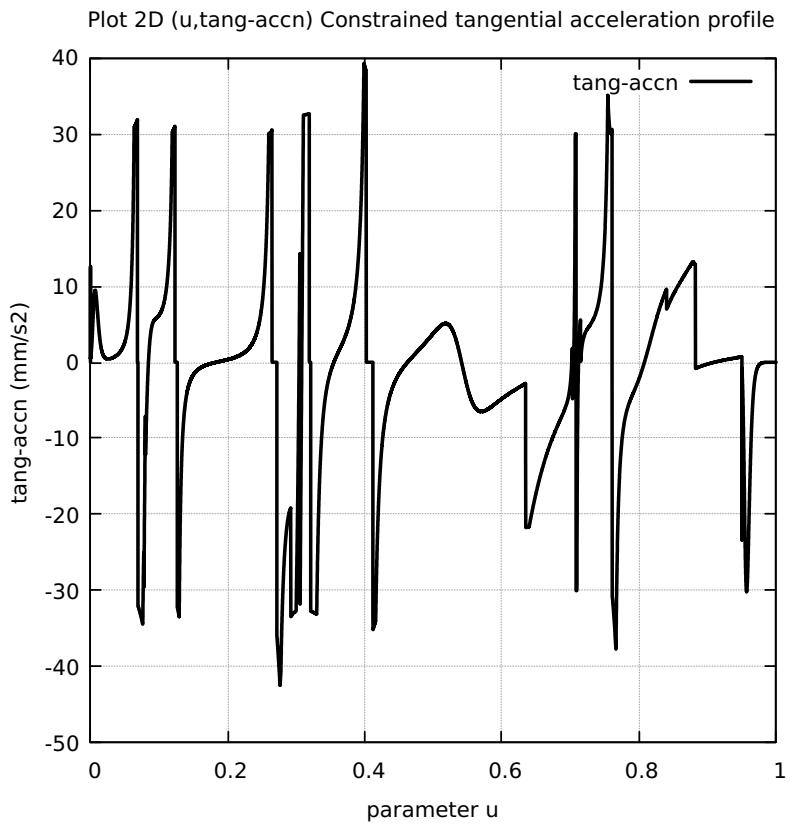


Figure 344: SnaHyp FC20 Nominal Rising S-Curve Profile

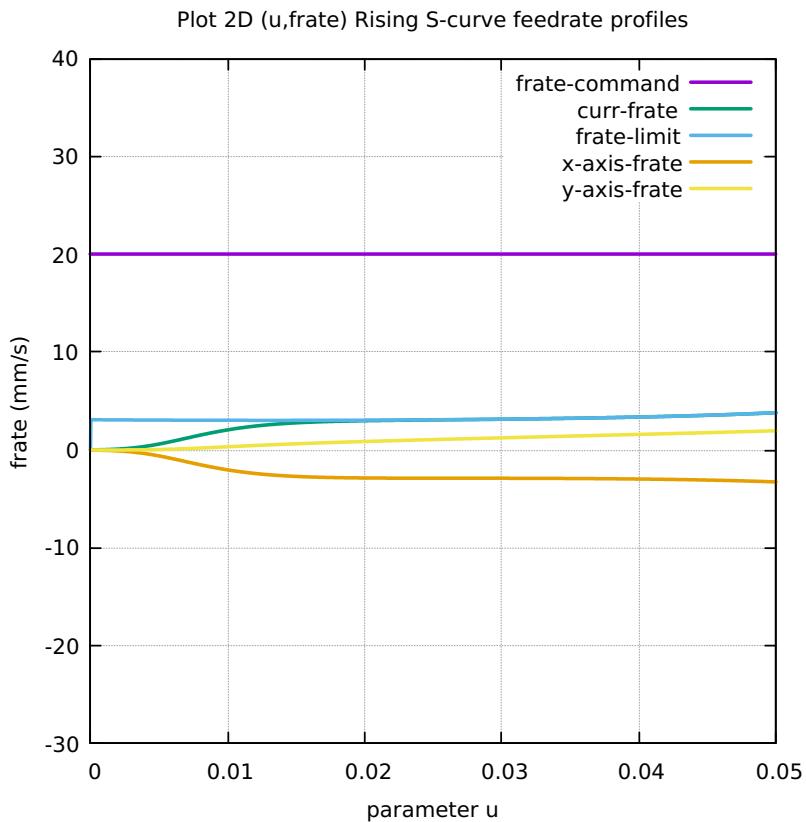


Figure 345: SnaHyp FC20 Nominal Falling S-Curve Profile

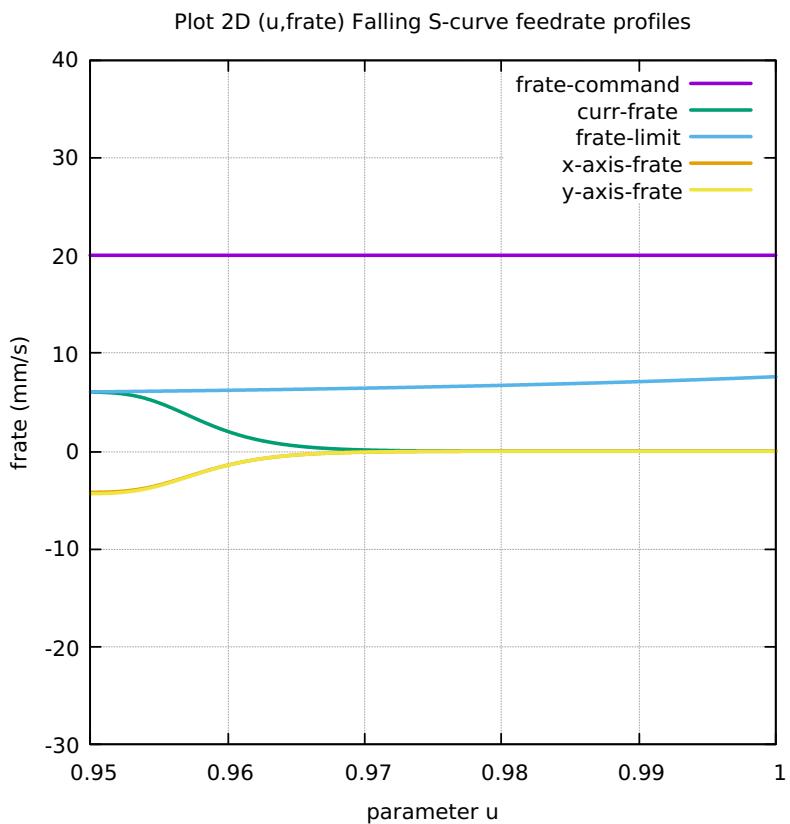


Figure 346: SnaHyp FC10 Colored Feedrate Profile data ngcode

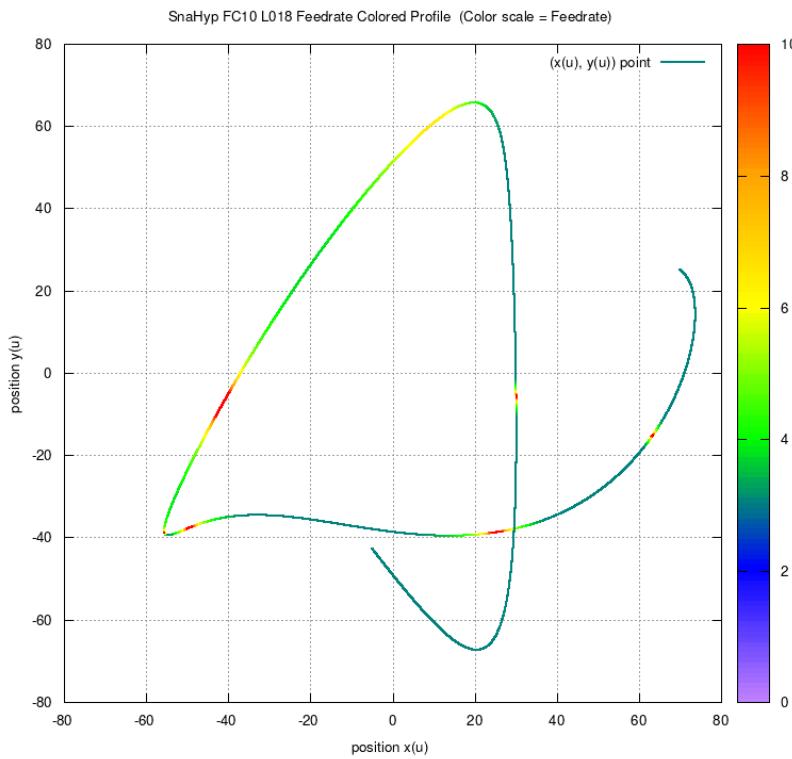


Figure 347: SnaHyp FC20 Colored Feedrate Profile data ngcode

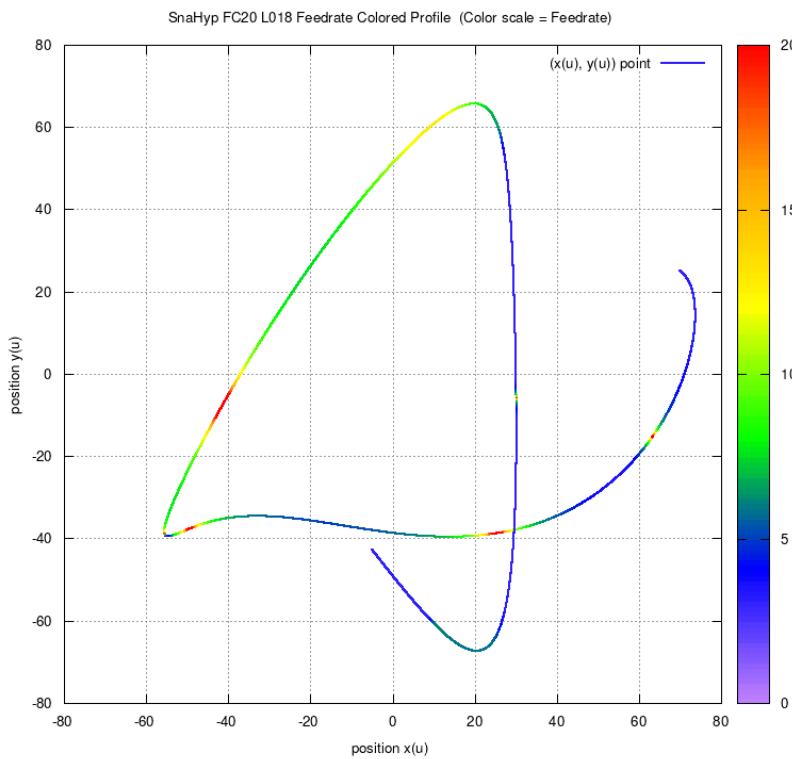


Figure 348: SnaHyp FC25 Colored Feedrate Profile data ngcode

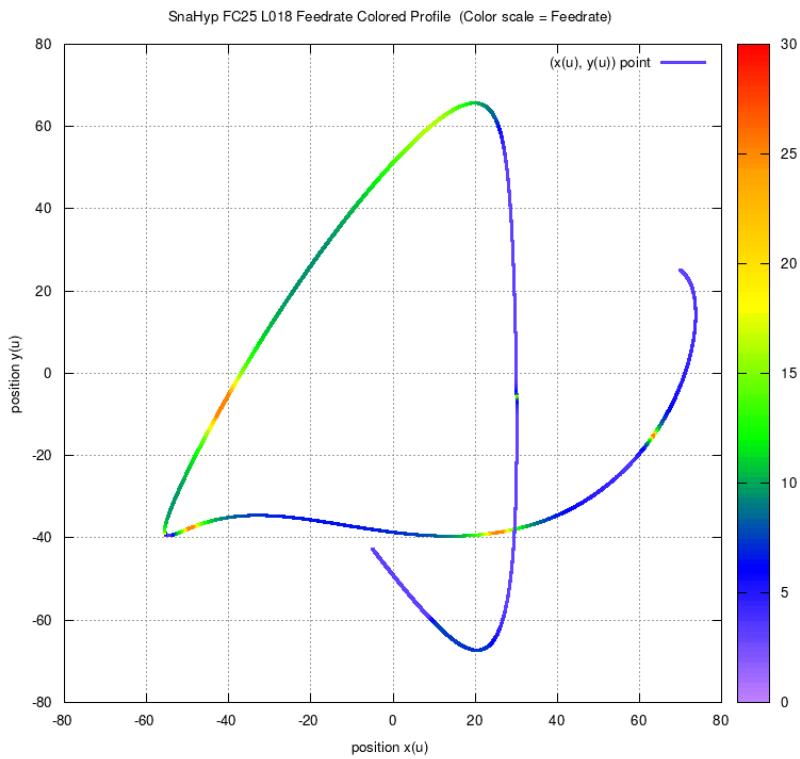


Figure 349: SnaHyp FC28 Colored Feedrate Profile data ngcode

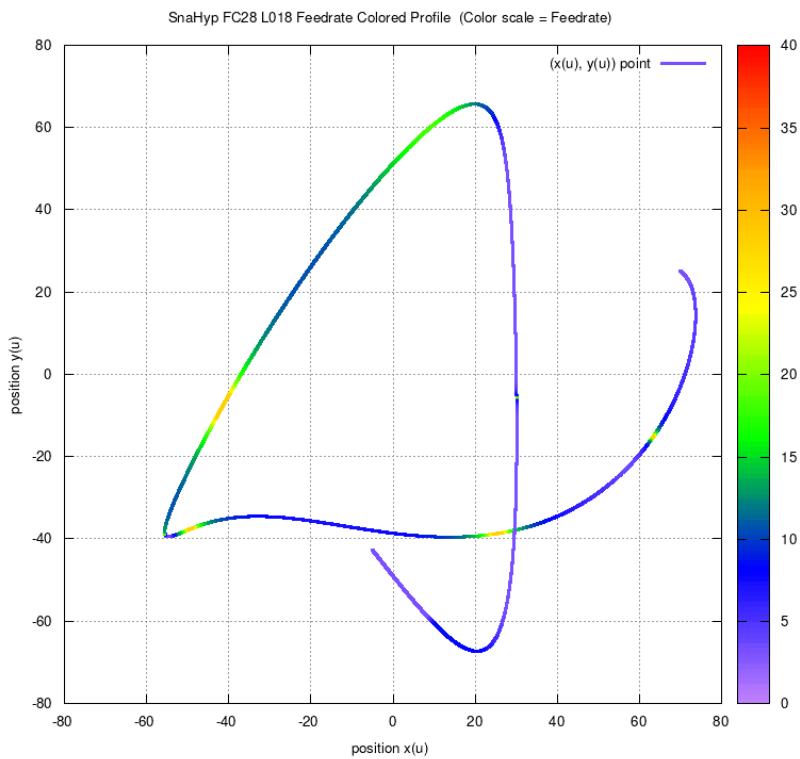


Figure 350: SnaHyp FC10 Tangential Acceleration

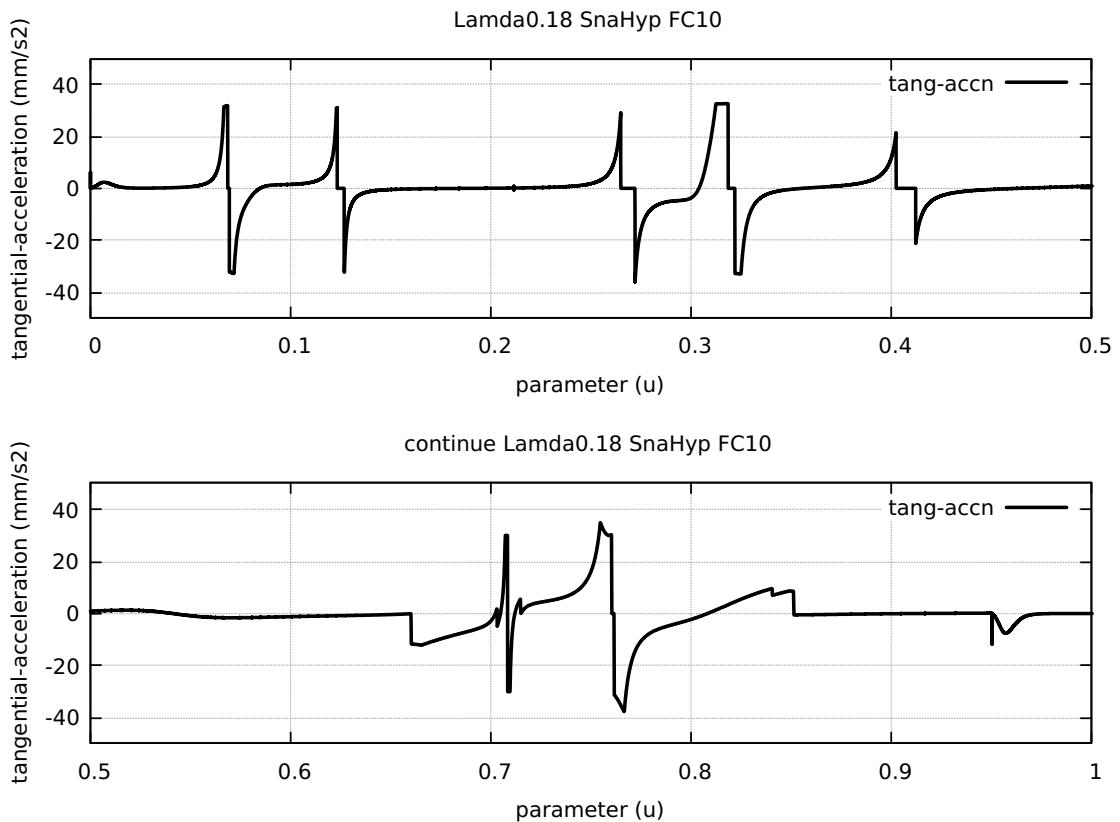


Figure 351: SnaHyp FC20 Tangential Acceleration

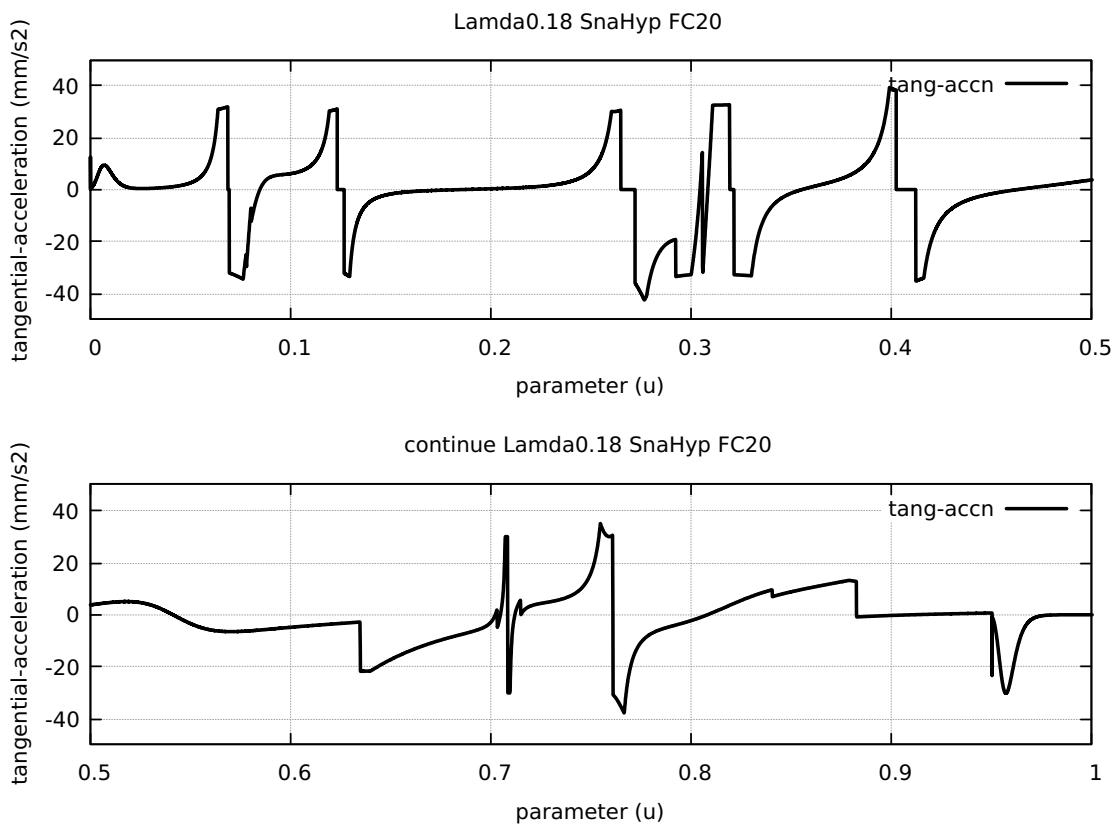


Figure 352: SnaHyp FC30 Tangential Acceleration

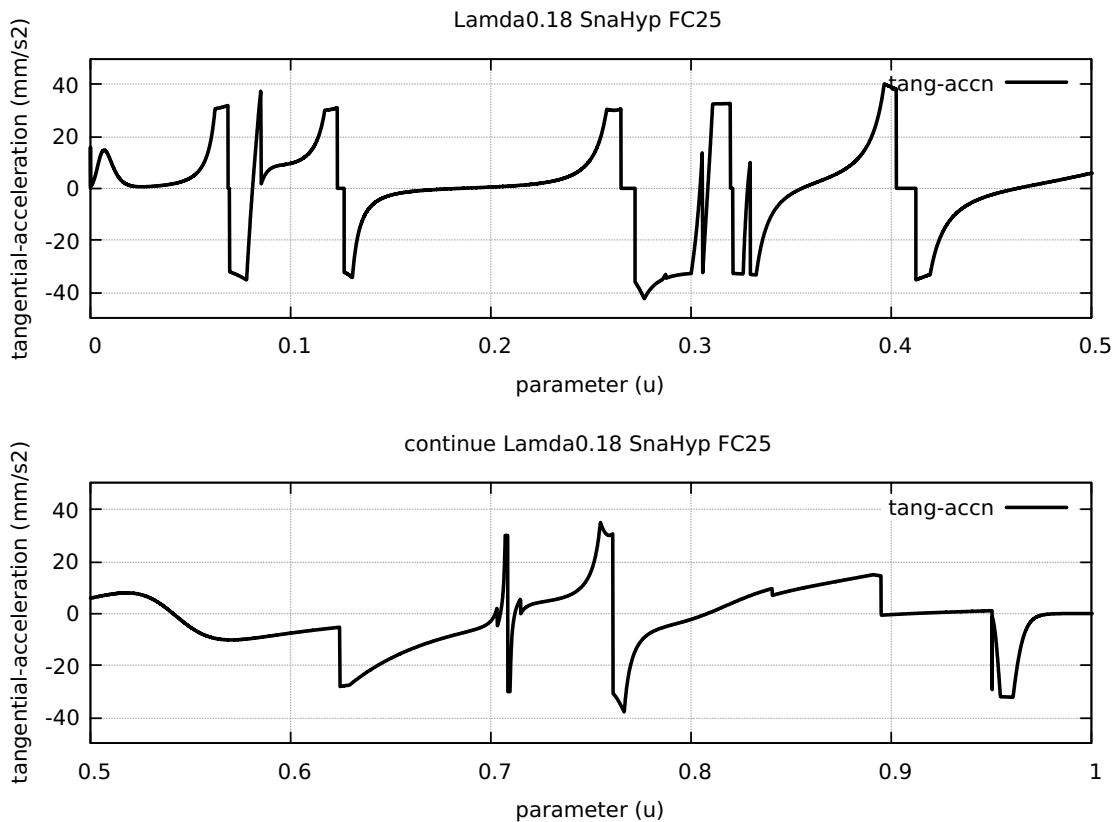


Figure 353: SnaHyp FC40 Tangential Acceleration

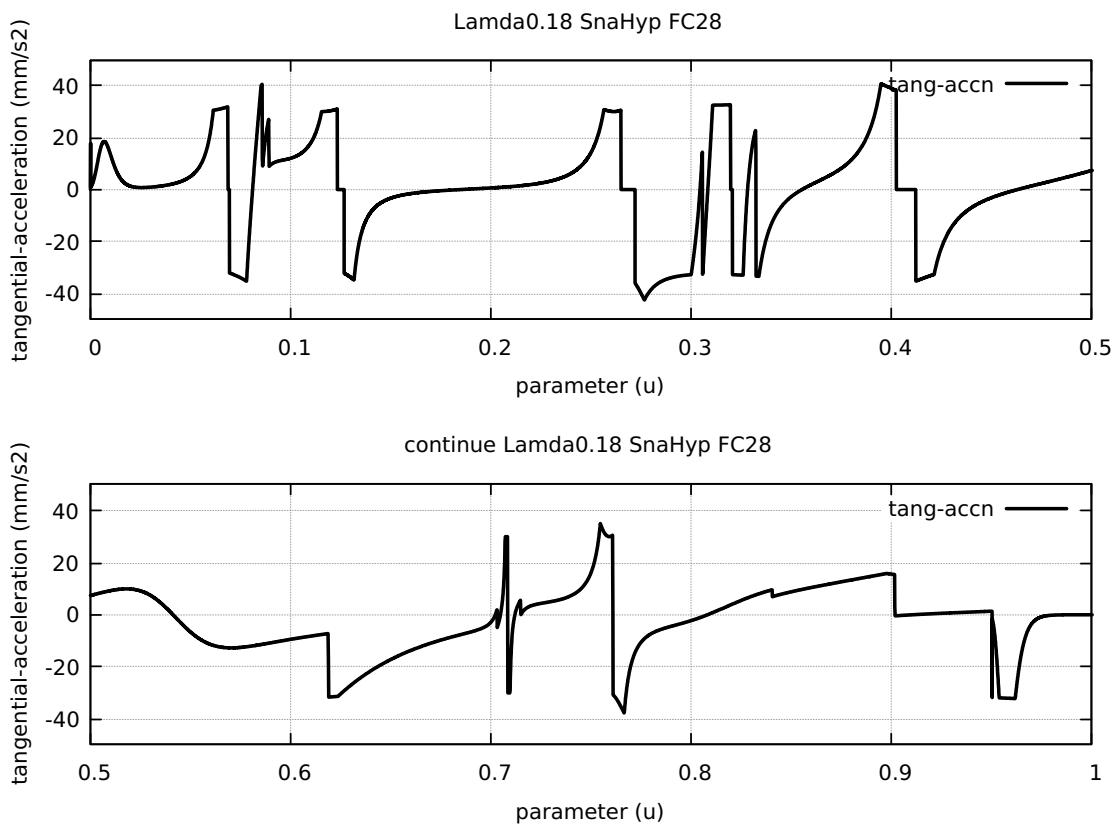


Figure 354: SnaHyp FC20 Nominal Separation NAL and NCL

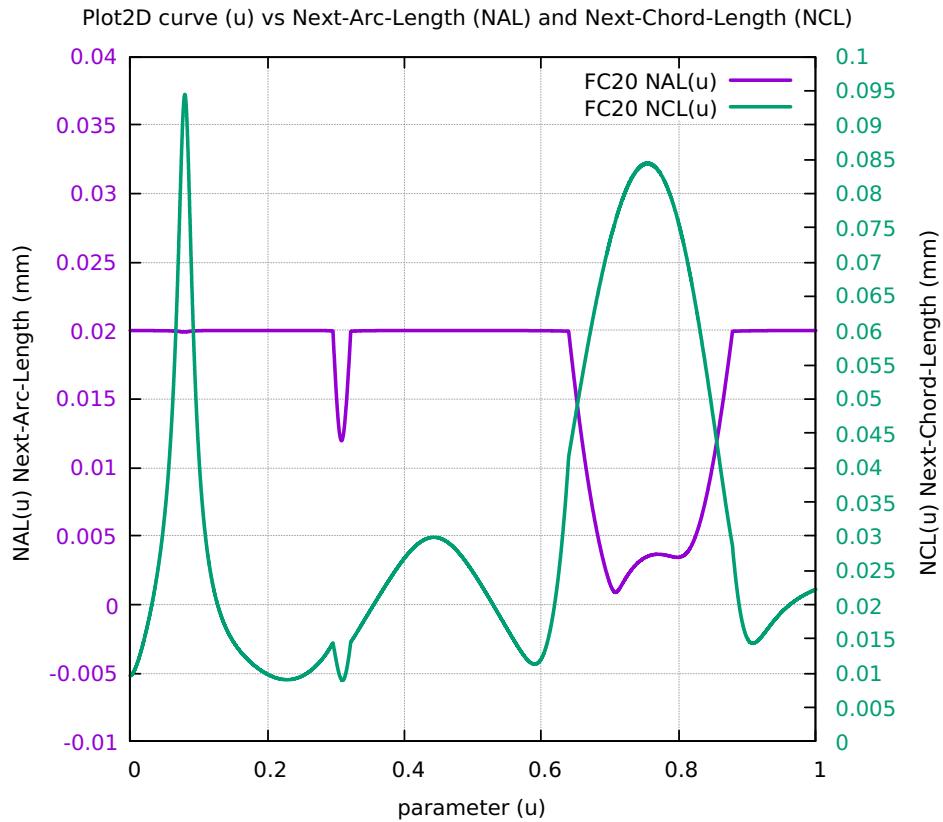


Figure 355: SnaHyp Difference SAL minus SCL for FC10 FC20 FC30 FC40

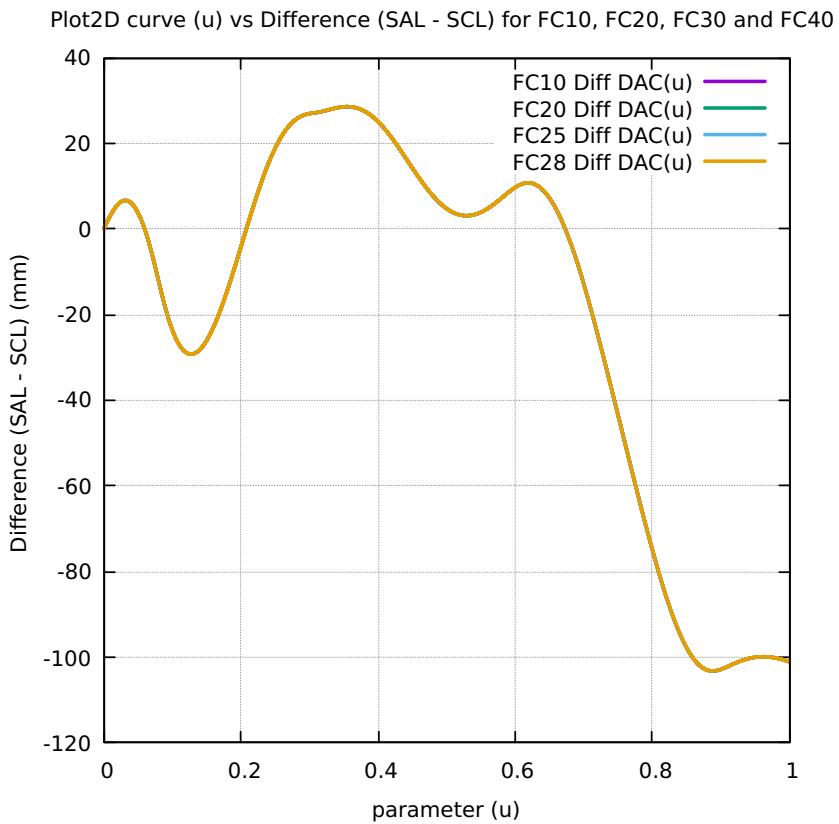


Figure 356: SnaHyp FC10 FrateCmd CurrFrate X-Frate Y-Frate

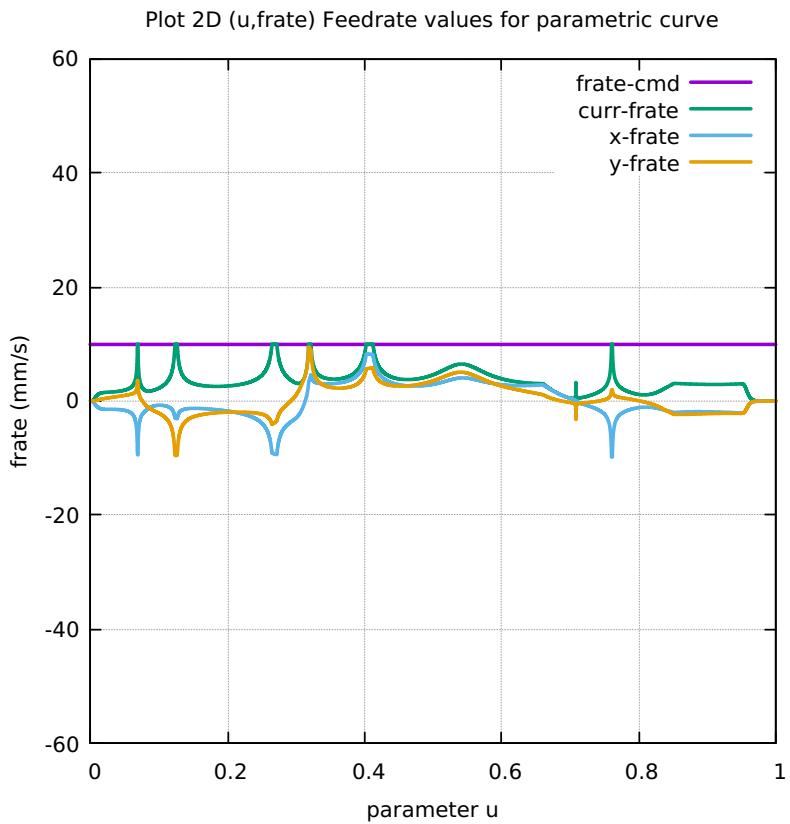


Figure 357: SnaHyp FC20 FrateCmd CurrFrate X-Frate Y-Frate

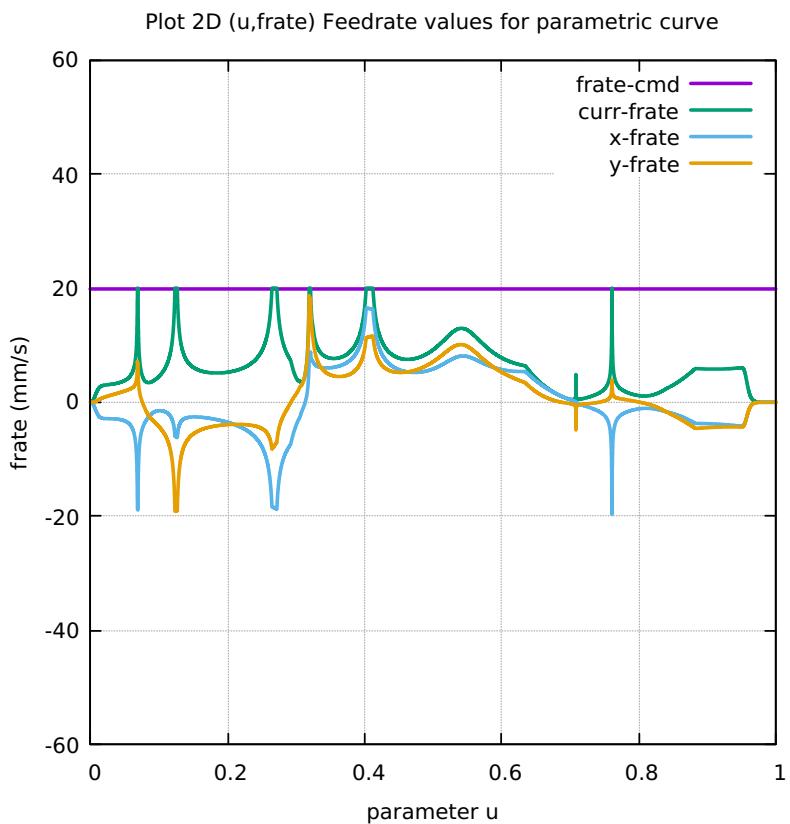


Figure 358: SnaHyp FC25 FrateCmd CurrFrate X-Frate Y-Frate

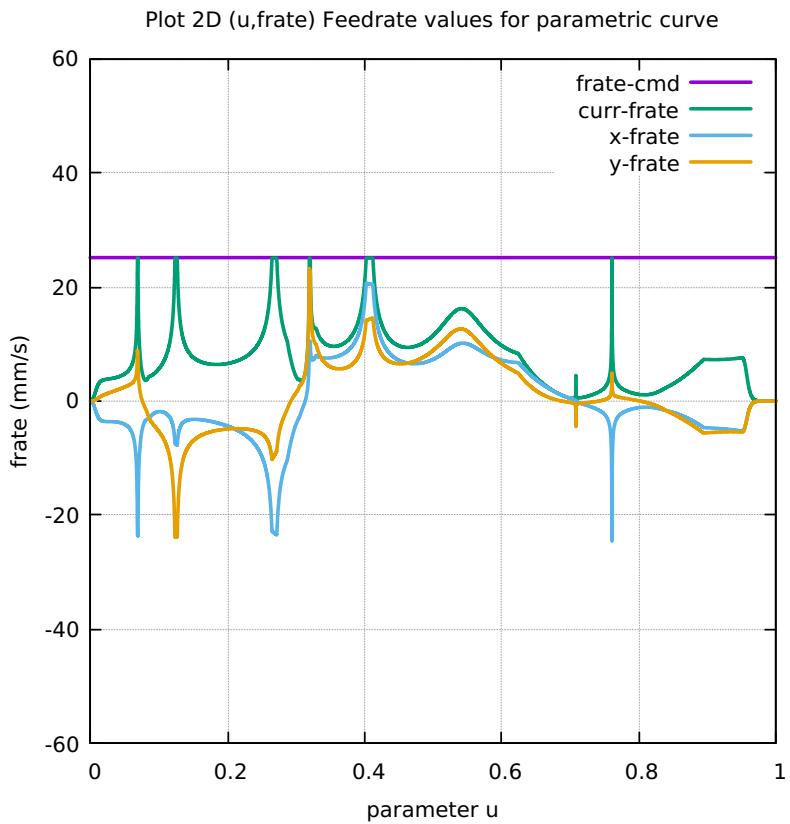


Figure 359: SnaHyp FC28 FrateCmd CurrFrate X-Frate Y-Frate

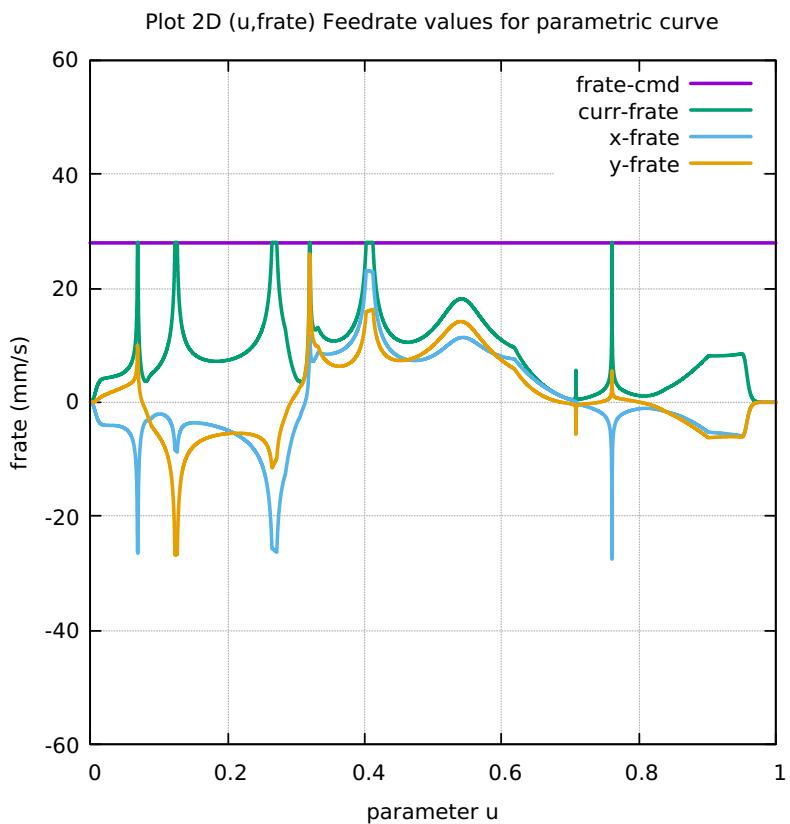


Figure 360: SnaHyp FC10 Four Components FeedrateLimit

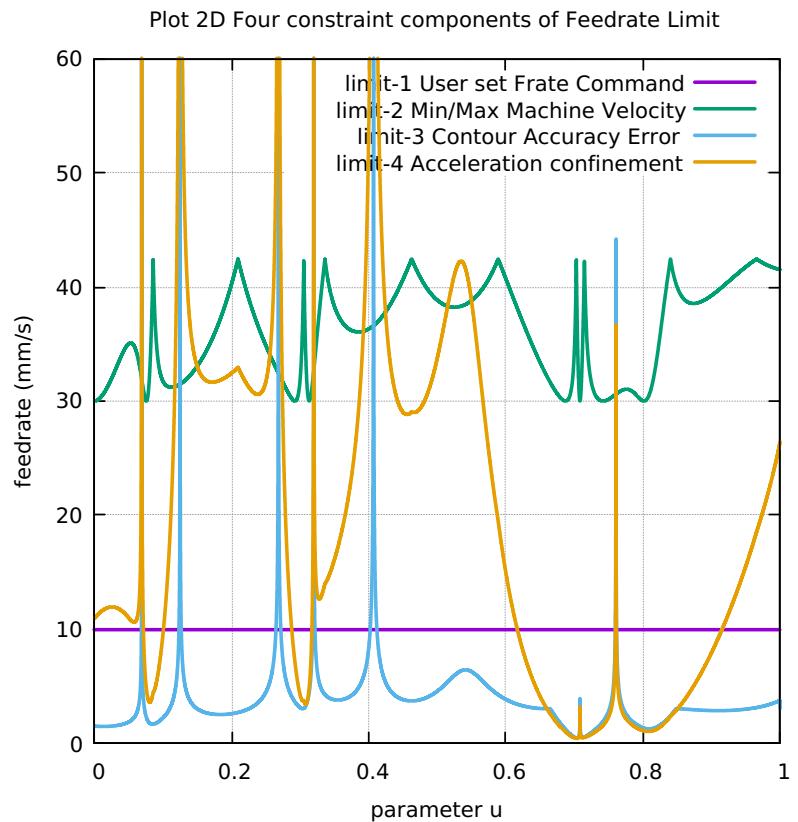


Figure 361: SnaHyp FC20 Four Components FeedrateLimit

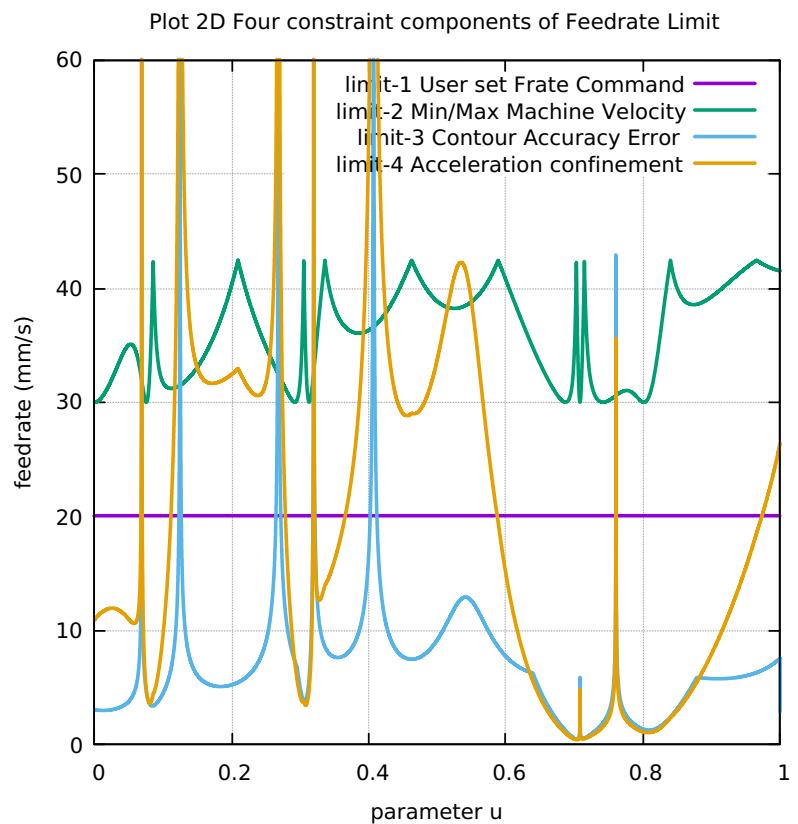


Figure 362: SnaHyp FC25 Four Components FeedrateLimit

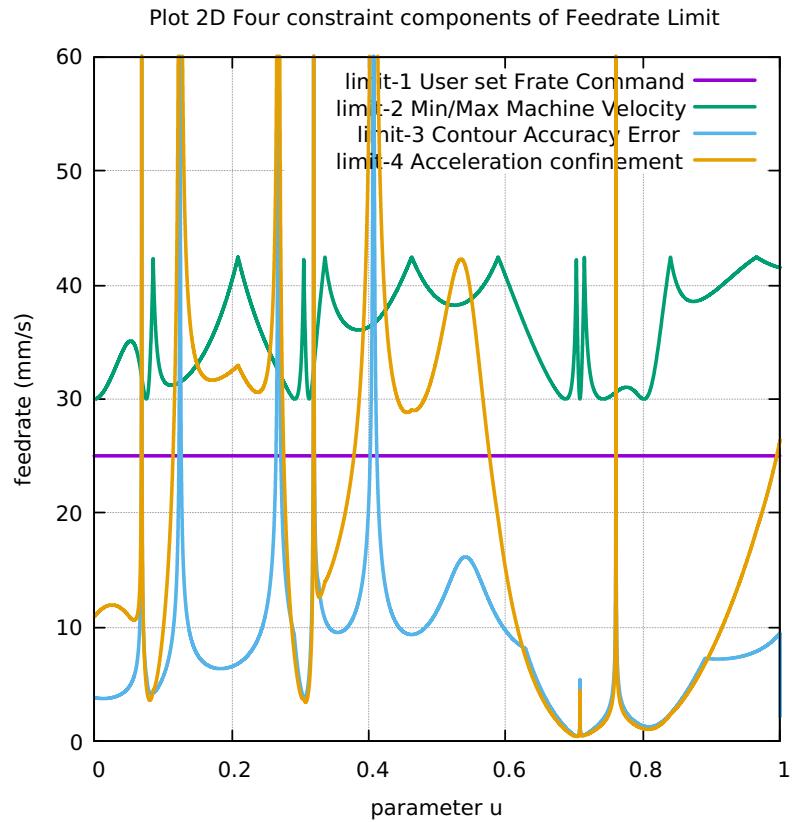


Figure 363: SnaHyp FC28 Four Components FeedrateLimit

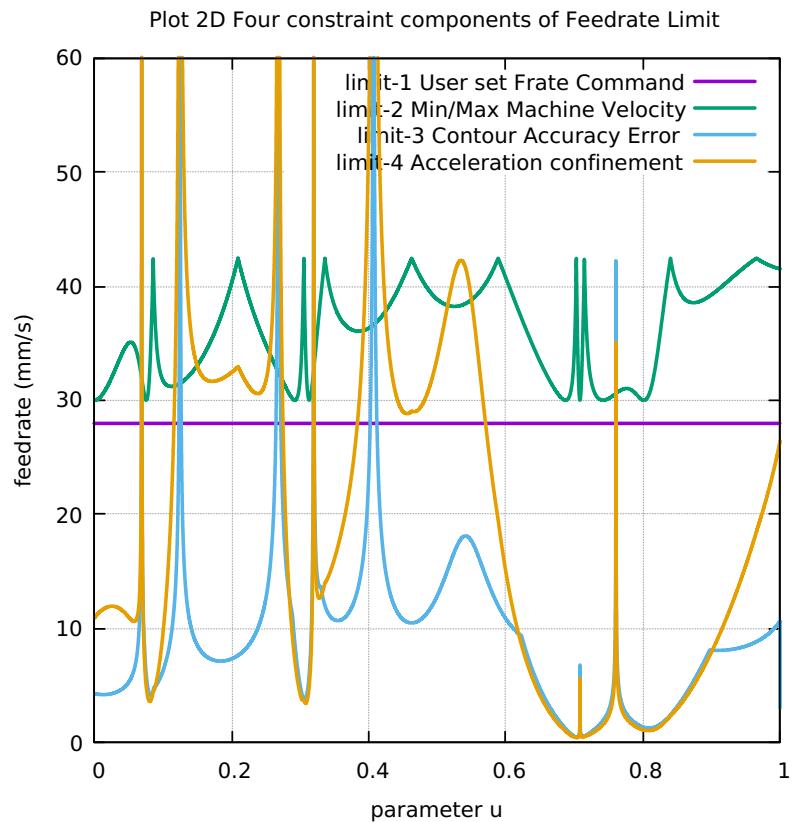


Figure 364: SnaHyp Histogram Points FC10 FC20 FC30 FC40

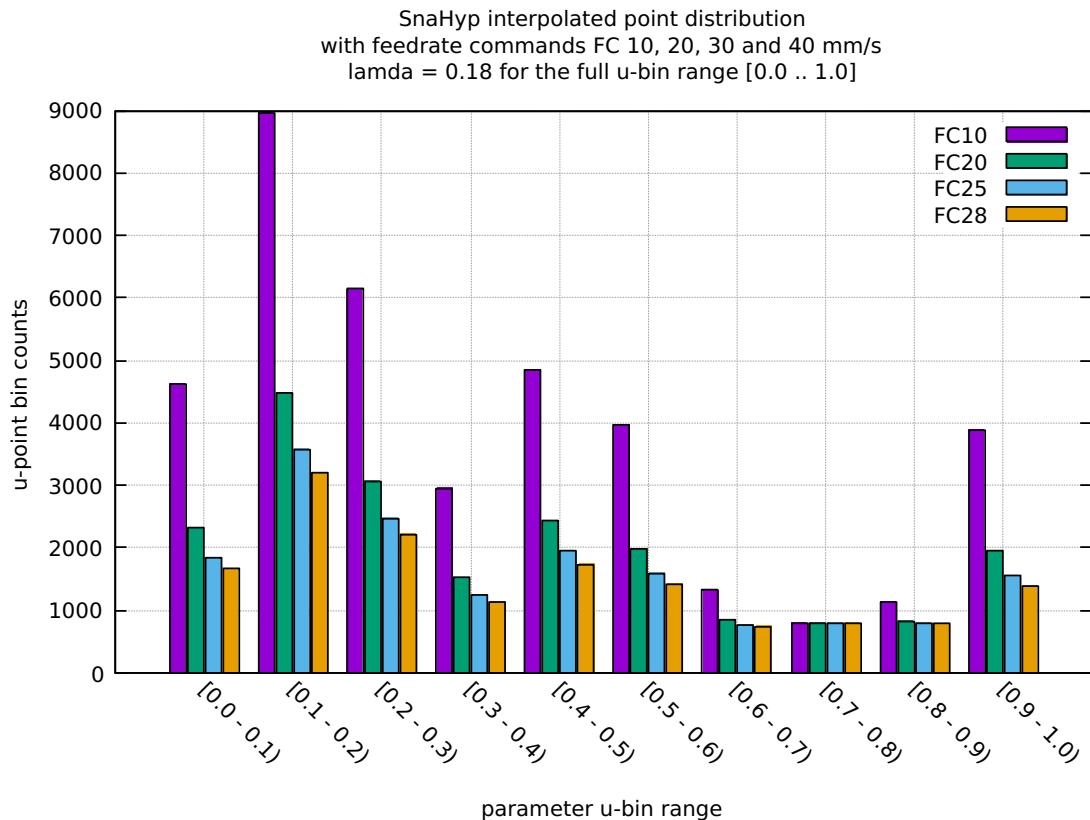


Table 20: SnaHyp Table distribution of interpolated points

BINS	FC10	FC20	FC25	FC28
0.0 - 0.1	4631	2317	1856	1666
0.1 - 0.2	8961	4480	3584	3200
0.2 - 0.3	6140	3074	2470	2213
0.3 - 0.4	2960	1526	1257	1146
0.4 - 0.5	4860	2431	1945	1736
0.5 - 0.6	3973	1987	1589	1420
0.6 - 0.7	1324	841	769	744
0.7 - 0.8	794	794	794	796
0.8 - 0.9	1141	828	798	791
0.9 - 1.0	3888	1945	1556	1390
Tot Counts	38672	20223	16618	15102

Table 21: SnaHyp Table FC10-20-30-40 Run Performance data

1	Curve Type	SNAHYP	SNAHYP	SNAHYP
2	User Feedrate Command FC(mm/s)	FC10	FC20	FC28**
3	User Lamda Acceleration Safety Factor	0.18	0.18	0.18
4	Total Interpolated Points (TIP)	38672	20223	16618
5	Total Sum-Chord-Error (SCE) (mm)	2.846873175820E-03	4.002975369043E-03	4.459254922025E-03
6	Ratio 1 = (SCE/TIP) = Chord-Error/Point	7.361778014067E-08	1.979515067275E-07	2.683549932012E-07
7	Total Sum-Arc-Length (SAL) (mm)	3.779474877648E+02	3.779592539969E+02	3.779667959094E+02
8	Total Sum-Chord-Length (SCL) (mm)	4.789870865777E+02	4.789986994127E+02	4.790063715425E+02
9	Difference = (SAL - SCL) (mm)	-1.010395988129E+02	-1.010394454158E+02	-1.010395756332E+02
10	Percentage Difference = (SAL - SCL)/SAL	-2.673376648446E+01	-2.673289365118E+01	-2.673223641934E+01
11	Ratio 2 = (SCE/SCL) = Chord Error/Chord-Length	5.943528031539E-06	8.356965006276E-06	9.309385400583E-06
12	Total Sum-Arc-Theta (SAT) (rad)	5.483921922253E+02	5.514331568816E+02	5.514225232921E+02
13	Total Sum-Arc-Area (SAA) (mm2)	2.706870610425E-01	2.760007501922E-01	2.777614243305E-01
14	Ratio 3 = (SAA/SCL) = Arc-Area/Chord-Length	5.943528031539E-06	8.356965006276E-06	9.309385400583E-06
15	Average-Chord-Error (ACE) (mm)	7.361778014067E-08	1.979515067275E-07	2.683549932012E-07
16	Average-Arc-Length (AAL) (mm)	9.773408698114E-03	1.869049817016E-02	2.274579020939E-02
17	Average-Chord-Length (ACL) (mm)	1.238620895704E-02	2.368700916886E-02	2.882628462072E-02
18	Average-Arc-Theta (AAT) (rad)	1.418096744913E-02	2.726897225208E-02	3.318424043402E-02
19	Average-Arc-Area (AAA) (mm2)	6.999742986798E-06	1.364853872971E-05	1.671549764280E-05
20	Algorithm actual runtime on computer (ART) (s)	18.694644926	16.525007035	16.49022576
				16.6538509