

# PhD Proposal Writeup

**A realtime and parallel look-ahead control and feedrate compensation strategy for CNC reference-pulse interpolation.**

**Faculty of Mechanical Engineering,**  
Universiti Malaysia Pahang (UMP),  
26600 Pekan, Pahang Darul Makmur,  
Malaysia.

PhD Program Registration Details		
1	Name of Student	Wan Ruslan bin W Yusoff
2	Student ID	PFD18001
3	National Reg. ID	560911-03-5067
4	Faculty	Faculty of Mechanical Engineering
5	Program	Doctor of Philosophy (PhD)
6	Field of Research	Mechatronics and System Design
7	Type of Study	Research
8	Mode of Study	Full Time
9	Registration Date	Tue, 03 April 2018
10	Supervisor	Dr. Fadhlur Rahman bin Mohd Romlay
12	External Advisor	Prof. Yashwant Prasad Singh
13	Document Date	June 17, 2023
14	Research Title	A realtime and parallel look-ahead control and feedrate compensation strategy for CNC reference-pulse interpolation
15	Contact EMail	wruslandr@gmail.com
16	Contact Mobile	6012-3218120

Reference: **Draft-44-Report-Latex-PhD-Proposal-WRY.tex**  
Date: **June 17, 2023**  
Version: **Draft-44**

# Contents

Cover Page	1
Abstract	1
Acknowledgement	1
Acronyms	1
Table of Contents	1
Contents	1
List of Figures	5
List of Tables	6
Listings	8
<b>1 Introduction</b>	<b>9</b>
1.1 Introduction	9
1.2 Types of CNC Machines	9
1.2.1 CNC Milling Machine	9
1.2.2 CNC Lathe Machine	10
1.2.3 CNC Router Machine	10
1.2.4 CNC 3D Printing	10
1.2.5 Recent Designs in CNC Systems	10
1.3 Basic architecture of a typical CNC system	11
1.4 Process Flow for a typical CNC system	12
1.5 Functional block diagram for a typical CNC system	13
1.5.1 CNC operations in CAD/CAM/CNC systems	14
1.5.2 Interpreter and interpolator	15
1.5.3 Interpolation Task	15
1.5.4 Starting point of CNC operations	15
1.5.5 Online versus Offline processing	15
1.5.6 Realtime processing	16
1.5.7 Parallel processing	17
1.5.8 Recent interpolation Methods	18
1.6 CNC System Hardware and Software	21
1.6.1 CNC Hardware Components	21
1.6.2 CNC Software Components	21
1.6.3 CNC Interpreter	22

1.6.4	CNC Interpolator . . . . .	23
1.6.5	CNC Signal Driver software . . . . .	27
1.6.6	CNC Device Driver hardware . . . . .	29
1.6.7	CNC Motor Control Driver hardware . . . . .	30
1.6.8	CNC Electric Motor hardware . . . . .	30
1.6.9	CNC Control Loop design . . . . .	30
1.6.10	Commercial CNC Control Software . . . . .	32
1.7	Research Motivations . . . . .	34
1.8	Scope of Research . . . . .	36
1.9	Proposed Research Title . . . . .	36
1.10	Expected knowledge contributions . . . . .	36
1.11	Summary on Introduction . . . . .	37
<b>2</b>	<b>Literature Survey</b>	<b>38</b>
2.1	Reviews - CNC State of the Art . . . . .	38
2.2	Overview of current issues in CNC . . . . .	38
2.3	G-Codes Standards . . . . .	38
2.3.1	NGC RS274D G-Codes . . . . .	39
2.3.2	NURBS G-Codes . . . . .	39
2.3.3	STEP-NC G-Codes . . . . .	40
2.4	Reference-Pulse interpolation . . . . .	40
2.5	Reference-Word interpolation . . . . .	41
2.6	NURBS CNC Interpolations . . . . .	42
2.7	Contouring control and Error Compensation . . . . .	44
2.8	Look-ahead control . . . . .	44
2.9	Feedrate Filtering . . . . .	45
2.10	Realtime and Parallel Computing . . . . .	46
2.11	AI Approaches in CNC Machining . . . . .	47
2.12	Commercial versus Open CNC systems . . . . .	47
2.13	Embedded devices in CNC systems . . . . .	47
2.14	Summary on Literature Survey . . . . .	48
<b>3</b>	<b>Research Methodology</b>	<b>49</b>
3.1	Research Objectives . . . . .	49
3.1.1	Research scope . . . . .	49
3.1.2	G-Codes Coverage . . . . .	49
3.1.3	Reference-Pulse Interpolation . . . . .	49
3.1.4	Look-ahead control . . . . .	50
3.1.5	Feedrate compensation . . . . .	50
3.1.6	Realtime and parallel execution . . . . .	50
3.1.7	Parallel execution strategies . . . . .	50
3.1.8	Extension to 5-axis interpolation . . . . .	51
3.2	Research Methodology . . . . .	51
3.2.1	Software engineering perspective . . . . .	51
3.2.2	Software engineering methods . . . . .	51
3.2.3	Linux, open source and free software . . . . .	52
3.2.4	Motion control devices . . . . .	52
3.2.5	Research Validation . . . . .	53
3.3	Summary on Research Methodology . . . . .	54

<b>4</b>	<b>Software Interpolator Design</b>	<b>55</b>
<b>5</b>	<b>Simulation Experiment</b>	<b>56</b>
5.1	Selected Parametric Equations . . . . .	56
5.1.1	Teardrop parametric equation . . . . .	57
5.1.2	Butterfly parametric equation . . . . .	58
5.1.3	Ellipse parametric equation . . . . .	59
5.1.4	Skewed-Astroid parametric equation . . . . .	60
5.1.5	Circle parametric equation . . . . .	61
5.1.6	AstEpi parametric equation . . . . .	62
5.1.7	Snailshell parametric equation . . . . .	63
5.1.8	SnaHyp parametric equation . . . . .	64
5.1.9	Ribbon-10L parametric equation . . . . .	65
5.1.10	Ribbon-100L parametric equation . . . . .	66
5.2	Total Interpolated Points . . . . .	67
5.2.1	Teardrop profile of interpolated points . . . . .	68
5.2.2	Butterfly profile of interpolated points . . . . .	69
5.2.3	Ellipse profile of interpolated points . . . . .	70
5.2.4	Skewed-Astroid profile of interpolated points . . . . .	71
5.2.5	Circle profile of interpolated points . . . . .	72
5.2.6	Astroid-Epicycloid profile of interpolated points . . . . .	73
5.2.7	Snailshell-Hypotrochoid profile of interpolated points . . . . .	74
5.2.8	Snailshell profile of interpolated points . . . . .	75
5.2.9	Ribbon-10L profile of interpolated points . . . . .	76
5.2.10	Ribbon-100L profile of interpolated points . . . . .	77
5.3	Experimental Run Results . . . . .	78
5.3.1	Teardrop and Butterfly Run Data . . . . .	79
5.3.2	Ellipse and Skewed-Astroid Run Data . . . . .	80
5.3.3	Circle and AstEpi Run Data . . . . .	81
5.3.4	Snailshell and SnaHyp Run Data . . . . .	82
5.3.5	Ribbon-10L and Ribbon-100L Run Data . . . . .	83
5.4	Results Feedrate Profile . . . . .	84
5.4.1	Teardrop FC20 u versus x-y-curr feedrate profile . . . . .	85
5.4.2	Teardrop FC20 x-y and colored feedrate profile . . . . .	86
5.4.3	Butterfly FC20 u versus x-y-curr feedrate profile . . . . .	87
5.4.4	Butterfly FC20 x-y and colored feedrate profile . . . . .	88
5.4.5	Ellipse FC20 u versus x-y-curr feedrate profile . . . . .	89
5.4.6	Ellipse FC20 x-y and colored feedrate profile . . . . .	90
5.4.7	Skewed-Astroid FC20 u versus x-y-curr feedrate profile . . . . .	91
5.4.8	Skewed-Astroid FC20 x-y and colored feedrate profile . . . . .	92
5.4.9	Circle FC20 u versus x-y-curr feedrate profile . . . . .	93
5.4.10	Circle FC20 x-y and colored feedrate profile . . . . .	94
5.4.11	AstEpi FC20 u versus x-y-curr feedrate profile . . . . .	95
5.4.12	AstEpi FC20 x-y and colored feedrate profile . . . . .	96
5.4.13	Snailshell FC20 u versus x-y-curr feedrate profile . . . . .	97
5.4.14	Snailshell FC20 x-y and colored feedrate profile . . . . .	98
5.4.15	SnaHyp FC20 u versus x-y-curr feedrate profile . . . . .	99
5.4.16	SnaHyp FC20 x-y and colored feedrate profile . . . . .	100
5.4.17	Ribbon-10L FC20 u versus x-y-curr feedrate profile . . . . .	101
5.4.18	Ribbon-10L FC20 x-y and colored feedrate profile . . . . .	102

5.4.19	Ribbon-100L FC20 u versus x-y-curr feedrate profile . . . . .	103
5.4.20	Ribbon-100L FC20 x-y and colored feedrate profile . . . . .	104
5.5	Error per unit length traversed . . . . .	105
5.5.1	FC10 - Error per unit length traversed . . . . .	106
5.5.2	FC20 - Error per unit length traversed . . . . .	107
5.5.3	FC25 - Error per unit length traversed . . . . .	108
5.5.4	FC30 - Error per unit length traversed . . . . .	109
5.5.5	FC40 - Error per unit length traversed . . . . .	110
5.6	Histogram of Interpolated Points . . . . .	111
5.6.1	Teardrop histogram of interpolated points . . . . .	112
5.6.2	Butterfly histogram of interpolated points . . . . .	113
5.6.3	Ellipse histogram of interpolated points . . . . .	114
5.6.4	Skewed-Astroid histogram of interpolated points . . . . .	115
5.6.5	Circle histogram of interpolated points . . . . .	116
5.6.6	AstEpi histogram of interpolated points . . . . .	117
5.6.7	Snailshell histogram of interpolated points . . . . .	118
5.6.8	SnaHyp histogram of interpolated points . . . . .	119
5.6.9	Ribbon-10L histogram of interpolated points . . . . .	120
5.6.10	Ribbon-100L histogram of interpolated points . . . . .	121
5.7	Experimental Validation . . . . .	122
<b>6</b>	<b>Analyses and Discussion</b>	<b>123</b>
<b>7</b>	<b>Conclusion</b>	<b>124</b>
<b>1</b>	<b>Appendices</b>	<b>125</b>

# List of Figures

1.1	Basic architecture of a typical CNC system.	11
1.2	CNC Servo and Spindle Driving Mechanism	11
1.3	Process Flow for a typical CNC system	12
1.4	Functional Block Diagram for CNC system	13

# List of Tables

1.1	CNC Terminology Part 1 of 2 . . . . .	19
1.2	CNC Terminology Part 2 of 2 . . . . .	20
5.1	Teardrop parametric equation and dimensions . . . . .	57
5.2	Butterfly parametric equation and dimensions . . . . .	58
5.3	Ellipse equation and dimensions . . . . .	59
5.4	Skewed-Astroid and dimensions . . . . .	60
5.5	Circle equation and dimensions . . . . .	61
5.6	Astepi equation and dimensions . . . . .	62
5.7	Snailshell equation and dimensions . . . . .	63
5.8	SnaHyp equation and dimensions . . . . .	64
5.9	Ribbon-10L equations and dimensions . . . . .	65
5.10	Ribbon-100L equation and dimensions . . . . .	66
5.11	Teardrop profile of interpolated points . . . . .	68
5.12	Butterfly profile of interpolated points . . . . .	69
5.13	Ellipse profile of interpolated points . . . . .	70
5.14	Skewed-Astroid profile of interpolated points . . . . .	71
5.15	Circle profile of interpolated points . . . . .	72
5.16	AstEpi profile of interpolated points . . . . .	73
5.17	SnaHyp profile of interpolated points . . . . .	74
5.18	Snailshell profile of interpolated points . . . . .	75
5.19	Ribbon-10L profile of interpolated points . . . . .	76
5.20	Ribbon-100L profile of interpolated points . . . . .	77
5.21	Teardrop and Butterfly Run Data . . . . .	79
5.22	Ellipse and Skewed-Astroid Run Data . . . . .	80
5.23	Circle and Astepi Run Data . . . . .	81
5.24	Snailshell and SnaHyp Run Data . . . . .	82
5.25	Ribbon-10L and Ribbon-100L Run Data . . . . .	83
5.26	Teardrop FC20 u versus x-y-curr feedrate profile . . . . .	85
5.27	Teardrop FC20 x-y and colored feedrate profile . . . . .	86
5.28	Butterfly FC20 u versus x-y-curr feedrate profile . . . . .	87
5.29	Butterfly FC20 x-y and colored feedrate profile . . . . .	88
5.30	Ellipse FC20 u versus x-y-curr feedrate profile . . . . .	89
5.31	Ellipse FC20 x-y and colored feedrate profile . . . . .	90
5.32	Skewed-Astroid FC20 u versus x-y-curr feedrate profile . . . . .	91
5.33	Skewed-Astroid FC20 x-y and colored feedrate profile . . . . .	92
5.34	Circle FC20 u versus x-y-curr feedrate profile . . . . .	93
5.35	Circle FC20 x-y and colored feedrate profile . . . . .	94
5.36	AstEpi FC20 u versus x-y-curr feedrate profile . . . . .	95
5.37	AstEpi FC20 x-y and colored feedrate profile . . . . .	96

5.38	Snailshell FC20 u versus x-y-curr feedrate profile . . . . .	97
5.39	Snailshell FC20 x-y and colored feedrate profile . . . . .	98
5.40	SnaHyp FC20 u versus x-y-curr feedrate profile . . . . .	99
5.41	SnaHyp FC20 x-y and colored feedrate profile . . . . .	100
5.42	Ribbon-10L FC20 u versus x-y-curr feedrate profile . . . . .	101
5.43	Ribbon-10L FC20 x-y and colored feedrate profile . . . . .	102
5.44	Ribbon-100L FC20 u versus x-y-curr feedrate profile . . . . .	103
5.45	Ribbon-100L FC20 x-y and colored feedrate profile . . . . .	104
5.46	FC10 - Error per unit length for all parametric curves . . . . .	106
5.47	FC10 - Error per unit length for all parametric curves . . . . .	106
5.48	FC20 - Error per unit length for all parametric curves . . . . .	107
5.49	FC20 - Error per unit length for all parametric curves . . . . .	107
5.50	FC25 - Error per unit length for all parametric curves . . . . .	108
5.51	FC25 - Error per unit length for all parametric curves . . . . .	108
5.52	FC30 - Error per unit length for all parametric curves . . . . .	109
5.53	FC30 - Error per unit length for all parametric curves . . . . .	109
5.54	FC40 - Error per unit length for all parametric curves . . . . .	110
5.55	FC40 - Error per unit length for all parametric curves . . . . .	110
5.56	Teardrop histogram of interpolated points . . . . .	112
5.57	Butterfly histogram of interpolated points . . . . .	113
5.58	Ellipse histogram of interpolated points . . . . .	114
5.59	Skewed-Astroid histogram of interpolated points . . . . .	115
5.60	Circle histogram of interpolated points . . . . .	116
5.61	AstEpi histogram of interpolated points . . . . .	117
5.62	Snailshell histogram of interpolated points . . . . .	118
5.63	SnaHyp histogram of interpolated points . . . . .	119
5.64	Ribbon-10L histogram of interpolated points . . . . .	120
5.65	Ribbon-100L histogram of interpolated points . . . . .	121



# Listings

# 1 Introduction

## 1.1 Introduction

With the availability of increased processing power and memory of personal computers, today Computer Numerical Control (CNC) interpolation is mostly carried out using computer software instead of electronic hardware devices with mathematical logic, for example, the Digital Differential Analyser (DDA) integrator. The basic theory and design of CNC systems, however, have remain the same since its beginning [?].

The two common types of CNC software interpolation techniques are Reference-Pulse Interpolation [?],[?], and Reference-Word (Sampled-Data) Interpolation [?], [?].

Over time, various algorithms have been proposed for CNC software interpolation. For example, techniques for Reference-Pulse interpolation include software version of the Digital Differential Analyser (SDDA), the Stairs Approximation and the Direct Search Interpolation.

Techniques for Reference-Word interpolation include Euler, Improved Euler, Taylor, Tustin and Improved Tustin interpolations.

In CNC machine operation, the function of interpolation is to generate coordinated movements to drive the separate axis-of-motions and/or rotation axes in order to achieve the desired path of the CNC cutting or milling tool relative to the workpiece. Essentially, interpolation generates the final reference commands that moves stepper or servo motor drives to produce the physical part that is to be machined. Reference commands mean commands to the CNC tool to trace/follow (refer to) the desired path or trajectory.

## 1.2 Types of CNC Machines

There are various types of CNC machines. The main categories are CNC Milling, CNC Lathe, CNC Router and CNC 3D Printing machines. There are many sub-categories, covering variations of CNC machines, and it is too numerous to list.

These machines share one common characteristic, that is, they all are numerically controlled by a computer, thus adopting the name Computer Numerical Control (CNC) machines. Most CNC machines cut or remove materials in order to produce a machined part. The exception is the CNC 3D Printing machine where materials are instead deposited layer by layer to produce a part.

### 1.2.1 CNC Milling Machine

CNC milling devices are the most widely used type of CNC machine. It is a machine that produces a work part by both drilling and cutting using a rotating cylindrical cutting tool. In a milling machine, the cutter is able to move along multiple axes simultaneously. It can create a part with a variety of shapes, slots and holes. To produce a part, the work-piece in

a milling machine is often moved across the milling tool (drill cutter) in different directions, like linear table movements and circular table rotations. Common milling machines offer from 3 to 5 motion axes. Please see Fig. ??, a link to the image of a typical CNC Milling machine.

### 1.2.2 CNC Lathe Machine

The CNC lathe machine is a machine that rotates a workpiece on a spindle to cut away excess material. The lathe uses cutting tools and drill bits with different diameters to apply on the workpiece and produce a symmetrical object. In a lathe machine, the cutter is not able to move along multiple axes. These machines can produce objects in a variety of shapes, cuts, and details based on cutting a rotating work piece. The two lathe configuration types are the CNC horizontal lathe and the CNC vertical lathe. Please see Fig. ??, a link to the image of a typical CNC Lathe machine.

### 1.2.3 CNC Router Machine

The CNC router machine is very similar in concept to a CNC milling machine. Instead of routing by hand, tool paths for the CNC router are controlled via computer numerical control system. The word rout means to "hollow out" an area in relatively hard material. Routers are normally used for cutting various hard materials, such as engraving wood, composites, aluminum, steel, ceramics, plastics, glass, and foams. The application of routers are mainly in woodworking, especially building cabinets, engraving decorations, doors or panels. Routers are not generally used for drilling or 3D model machining. Please see Fig. ??, a link to the image of a typical CNC Routing machine.

### 1.2.4 CNC 3D Printing

The CNC 3D Printing machine is a machine in which material is deposited, joined or solidified under computer control to create a three-dimensional object. Many different technologies are used in the 3D printing process, the most common being the fused deposit modeling (FDM) method. The machine uses inkjet-technology nozzles to apply a specialized thick liquid or powder to form each new layer. Common materials used in 3D printing are thick waxes and plastic polymers. Please see Fig. ??, a link to the image of a typical CNC 3D Printing machine.

### 1.2.5 Recent Designs in CNC Systems

On designs in CNC machine architectures, recent developments in CNC systems include, for example, designs with the introduction of more motion axes (up to 9 axes), cutting under water or fluid immersion environments, new sensors, smart computing, high-speed machining, energy efficient techniques, artificial intelligence control, wastage reduction in materials removal, process monitoring, machining management operations, modular software construction, and human-machine interface.

On designs in CNC machining control, new developments include design, for example, in software algorithms and methods, for path motion smoothness, for machining speed control, for addressing acceleration and deceleration effects, for 2D path error reduction, for 3D contouring error control, for compensating tool jerks, vibrations, heating and friction effects.

### 1.3 Basic architecture of a typical CNC system

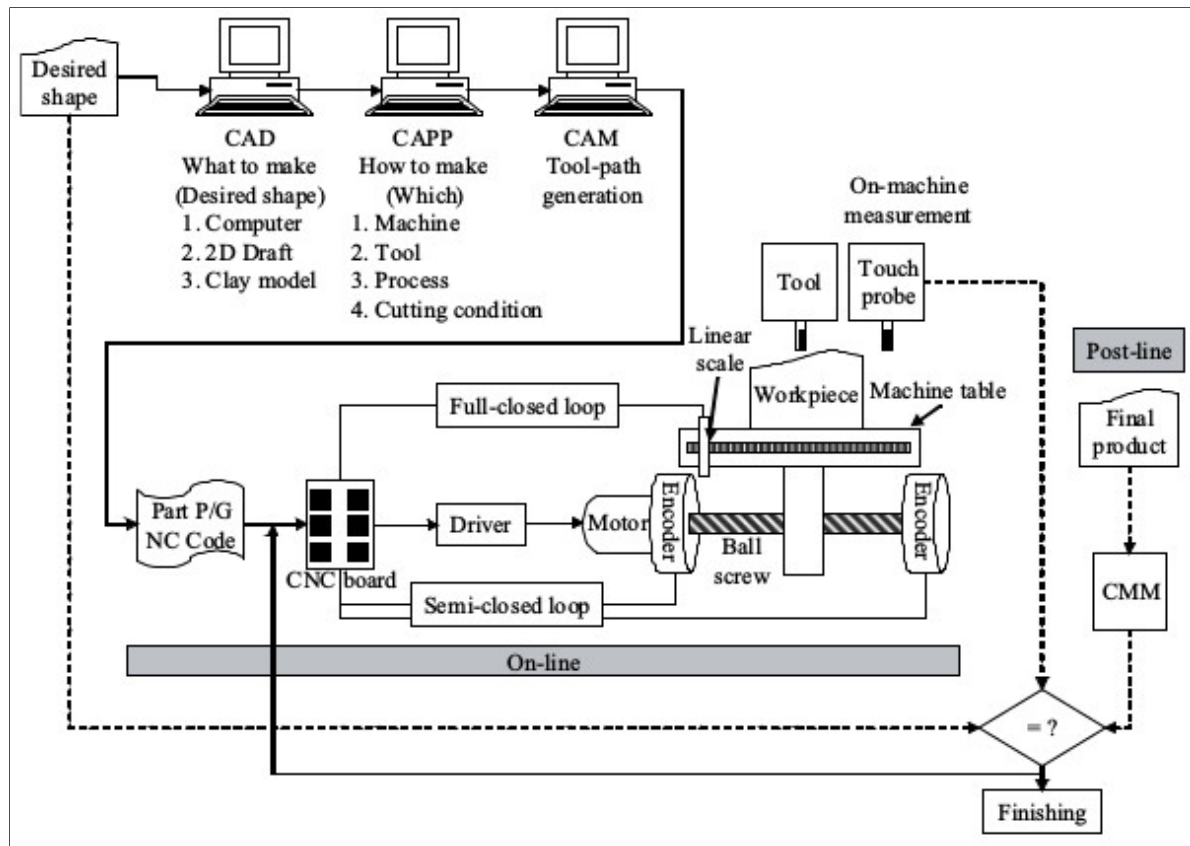


Figure 1.1: Basic architecture of a typical CNC system.

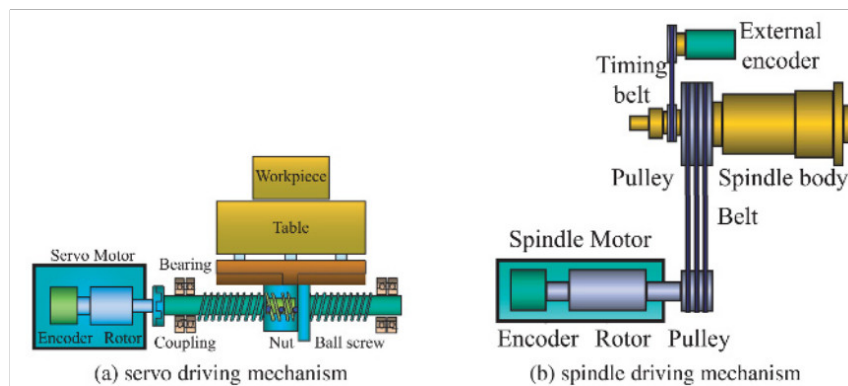


Figure 1.2: CNC Servo and Spindle Driving Mechanism

## 1.4 Process Flow for a typical CNC system

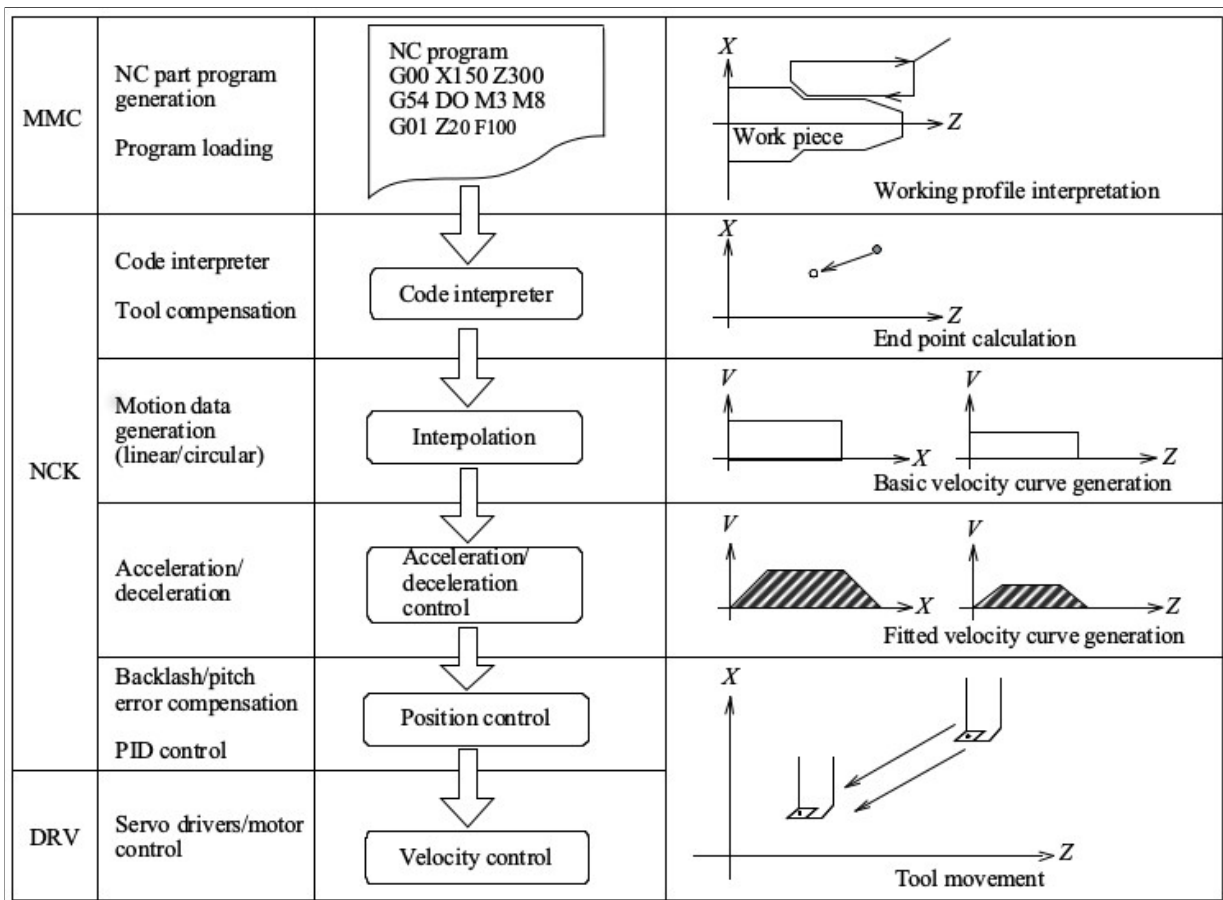


Figure 1.3: Process Flow for a typical CNC system

### Legend

1. MMC Man-Machine Control
2. NCK Numerical Control Kernel
3. DRV Driver Module
4. NC Numerical Control
5. PID Proportional Integrative Derivative

## 1.5 Functional block diagram for a typical CNC system

CNC manufacturing operations involve the generation of reference signals describing the geometrical parts to be machined and the control of the machine. The control is 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.

The figure below shows the functional block diagram for a typical CNC system. The meanings of terms used are described in Table [ 1.1] and Table [ 1.2].

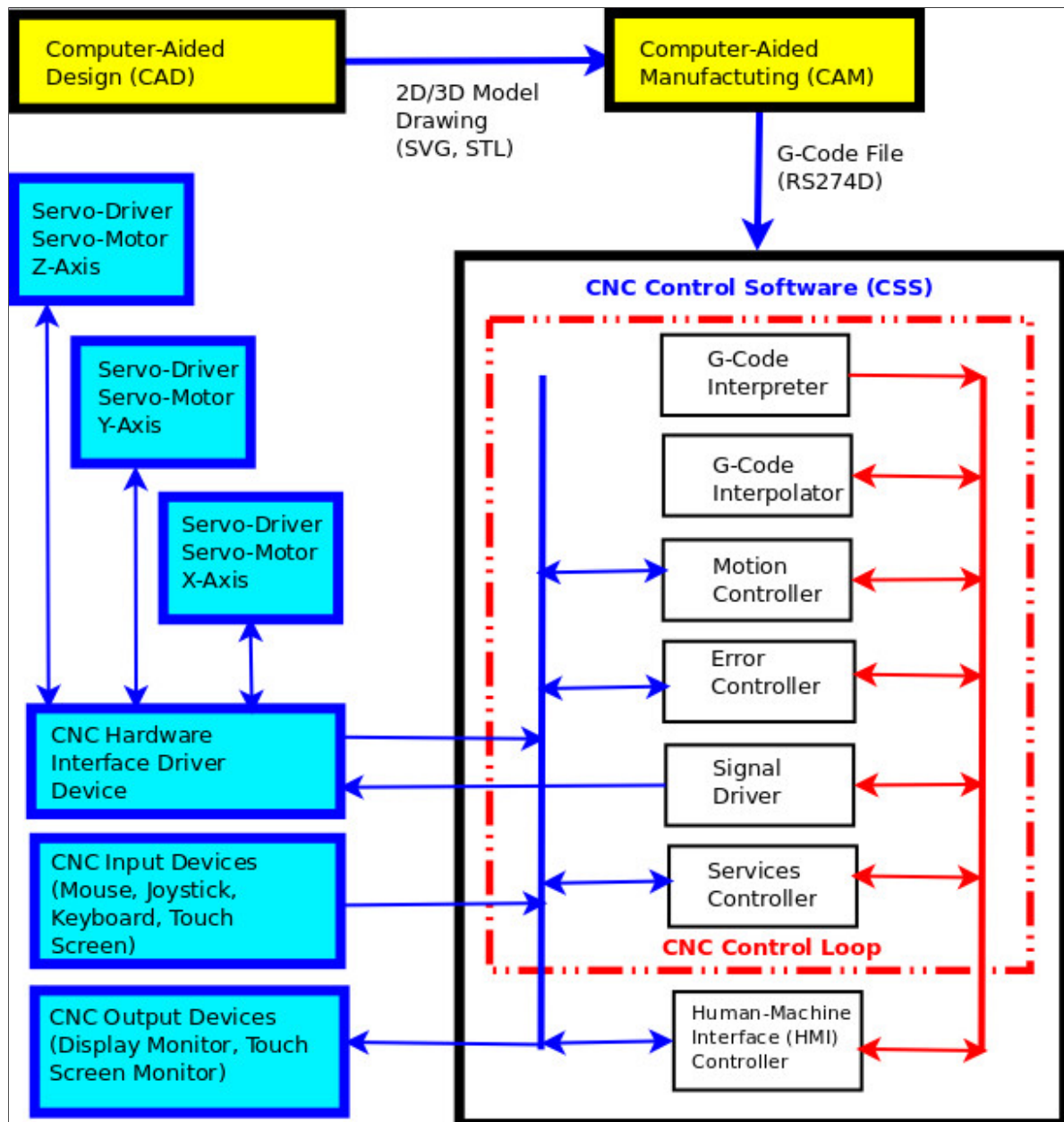


Figure 1.4: Functional Block Diagram for CNC system

### 1.5.1 CNC operations in CAD/CAM/CNC systems

The overall CNC operations in parts manufacturing using CAD/CAM/CNC systems can be described as follows.

First, the part geometry to be machined is generated by a Computer Aided Design (CAD) system that produces a 2D or 3D geometrical or model drawing.

Next, the Computer Aided Manufacturing (CAM) system converts this geometrical drawing into a part program called its G-Code. This G-Code file consists of a list of motion and machine service commands that provide specific step-by-step instructions for the CNC machine to manufacture the part. The G-Code part program generated by the CAM system is governed by international standards. This standardization ensures that different CAD drawings can be converted to a standardized G-Code by different CAM systems. Thus, different CNC manufacturers can machine the common G-Code and produce identical machine parts.

Next, a CNC interpreter program sequentially reads and interprets this G-Code commands, identifying whether the instruction is about tool movement processing, tool feedrate processing or machine function. For example, the prefix G in the part program instruction (G-Code) is for tool movement like a linear move or a circular arc move. This instruction moves the tool along the translational axes or along the rotary axes for some specified geometric path. The prefix M instruction is for controlling machine service functions like the start and stop of the tool cutting spindle. The prefix F instruction is for setting feedrates or speeds of tool movements along the translational axes or the rotary axes.

The next activity after CNC G-Code interpretation is CNC interpolation processing. CNC software interpolations cover linear, parametric and curve path contour commands provided in the G-Codes. The task of CNC G-Code interpolation is to process the interpreted G-Code and generate the final reference commands (referring to electrical signals) that drive the CNC tool along a pre-determined machining path. This contour path is normally constructed from a combination of linear and circular path segments. For example, interpolation generates reference commands that drive stepper or servo motors such that the required rotary or translational motions occur in a coordinated manner along the different axes simultaneously in tracing the desired contour path. In the process of CNC interpolation, the CNC Error Controller and CNC Motion Controller programs are executed based on to specific computation algorithms to achieve tracing accuracy along the desired contour path. These error-corrected or error-compensated outputs become machine reference commands (referring to the path) that drive the CNC motors.

The machine reference commands basically form a list of software machine instructions, saved in a CNC Signal File. The machine instructions in the signal file are read by the CNC Signal Driver in the CNC control loop and then transmitted to the CNC Hardware Interface Driver, which in turn, converts the software machine instructions that finally generates electrical pulses that drive the CNC electric motors.

The machine part to be manufactured is considered finished when the execution of the full list of reference commands completes and the CNC control loop exits.



### 1.5.2 Interpreter and interpolator

The interpreter and the interpolator programs are specific to a particular CNC machine. The programs are different across different CNC machines due to differences, for example, in machine hardware configurations, in number of motion axes, in functions of cutting tools, in machine service functions, in G-Code standards adoption, and so on. It is therefore, standard practice that most hardware manufacturers create their own proprietary interpreter and interpolator programs for their CNC machines. These program codes are not public.

Based on the above reason, in this research project, the development of our own interpreter and interpolator programs is our primary goal.

### 1.5.3 Interpolation Task

In CNC interpolation, it is important to differentiate between CAM model interpolation and CNC G-Code interpolation to avoid confusion. The CAM interpolation process converts 2D/3D model drawings to produce G-Codes files, whereas the CNC interpolation process converts G-Codes files into electrical signals to drive CNC motors.

Most advanced CAM software are commercial, not cost-free like many open-source type software. They are written by experts with well grounded knowledge in the mathematics of geometrical model representations for 2D and 3D objects. We do not want to be CAM designers, thus we leave CAM model interpolation to the experts.

For the above reasons, in this research project, CNC G-Code software interpolation is our only concern. CAM interpolation is not in our scope.

### 1.5.4 Starting point of CNC operations

Even though the CAD/CAM portion is not considered a component part of the CNC system, it plays an important initial role in the generation of G-Codes files. A high end and full-featured CAD/CAM system can produce G-Codes of different quality and contain different depths of manufacturing information. These sophisticated CAD/CAM applications can sometimes generate G-Codes complying to different G-Code standards.

In this research project, we consider the start of CNC machine operation as beginning from G-Code processing stage, and not from the CAD/CAM stages. In addition, we will only use G-Code files conforming to the RS274-D NGC standard. Other G-Code standards are not in our scope. We will discuss this further in the section on literature review.

### 1.5.5 Online versus Offline processing

In general, CNC Interpretation and CNC Interpolation can be conducted as online tasks, as offline tasks, or as a combination of both. In online mode, both interpretation and interpolation tasks are computed in memory and then fed into the CNC control loop while



the loop is running. In offline mode, the interpretation and interpolation are pre-computed, stored in a file or memory and later fed to the CNC control loop.

The term realtime processing used synonymously to mean online processing in common CNC literature, is not technically correct. In software engineering, the terms are conceptually unrelated. We clarify this in the next section.

In this research project, we will implement and compare performances of both online and offline options.

### 1.5.6 Realtime processing

In general usage, people used the word "realtime" as processing at the current time instance. The layman use of the term realtime in communications is very different from its technical use in software. We illustrate the difference as follows:

In software terms, the definition of realtime processing is the execution of a process that meets its time deadline. There are strictly two(2) time deadlines for any event: the start deadline and the end deadline.

As an example, consider two event deadlines for the safety airbag in an car. The event deadline to start opening the airbag is **after** 0.20 seconds from time of impact, and the event deadline for reaching a fully blown airbag is **before** 0.50 seconds from time of impact. It means **both events** must occur within the specific duration.

The two events are considered realtime events if they both strictly meet their deadlines. This is the technical criteria for being realtime. Opening the air bag before 0.20 seconds (too early) means missing the deadline. Also reaching a full blown air bag after 0.50 seconds (too late) also means missing its deadline. In both cases, the car driver may already be dead.

In this case, the safety airbag is designed such that it must open only after 0.20 seconds of impact and must get to full blown state before 0.50 seconds from time of impact.

If the airbag reaches a partially blown state at 0.50 seconds, the effectiveness of the safety airbag may be compromised, meaning, the design cannot guarantee the full protection of the car driver.

In any field of engineering design (including software programming), there is no such thing as an event to occur precisely at an exact point in time. This is unrealistic because there is no such thing as instantaneous in electronic or mechanical systems. Any event can only occur within some time duration, no matter how short the duration (for example, in microsecond or nanosecond range). It could be somewhere in the nanosecond range for timing control of ballistic missiles, whereas it is typically in the order of milliseconds, even for the most advanced CNC systems. Realtime compliance is very important in high performance time processing. Finally we state the technical definition: A realtime event must precisely occur within its specified duration, that is between its start and end deadlines.

In this research project, realtime shall mean the software event or process must comply to the technical definition of realtime above. This will be achieved only when program codes are written using realtime capable programming languages, and the codes are executed on Real Time Operating System (RTOS) platforms. For correct performance, CNC processing events typically runs in the order of 10 milliseconds. We will discuss this further in the section on literature review.

### 1.5.7 Parallel processing

Similarly, people have used the word "concurrent" as being synonymous with "parallel in time". In software terminology, we have to differentiate the layman use of the word parallel from its actual technical usage. We illustrate the difference between concurrent and parallel processes in software as follows:

Consider an example of a process where a person takes a jog and the jogging process is considered finished when the jogger reaches a specific destination. The jogger starts running and then along the way the jogger stops momentarily, to properly tie his loose shoe laces which took a few moments. During this short duration of tying the shoe lace, we consider the process of "shoe tying" and the process of "jogging" together as occurring concurrently.

Even though there is only a pause in the actual jogging, the jogging process is still considered ongoing because the destination has not yet being reached. When tightening the shoe laces in finished, the jogger resumes running until he reaches the destination, and then only the jogging process is considered finished. This is about concurrency.

All General Purpose Operating Systems (GPOS) platforms run their background processes concurrently. GPOS platforms are considered multi-tasking and time-sharing systems. Note that running in realtime (within start and end timing deadlines) and running in parallel are two separate concepts.

Next we explain about parallel processing. When two running processes or threads are executed on two separate CPUs independently at the same time, we say that the processes are truly running in parallel, meaning, overlapping in time. This is opposite to a sequential execution where one process must finish before another process can start. In parallel execution, processes truly run simultaneously or parallel in time.

In this research project, parallel shall mean simultaneously or overlapping in time. We shall implement program codes that run both in realtime and in parallel. We have successfully executed this before. We will discuss this further in the section on literature review.

### 1.5.8 Recent interpolation Methods

Recent techniques for CNC interpolation methods include advanced functions like look-ahead function, feedrate filtering and Non-Uniform Rational B-Splines (NURBS) interpolation. We will describe them in the chapter on literature review.

For this research project, we will study different interpolation methods. We propose the look-ahead control and feedrate compensation methods for realtime and parallel CNC interpolation.

CNC Terminology Part 1 of 2		
1,2	S/W and H/W	Software and Hardware
3	CNC G-Code File	The S/W file that is generated by a CAM application (including CAM interpolation) based on model or drawing files like DXF, STL (generated from CAD) and so on. This G-Code file, like RS-274D or STEP-NC file, contains instructions like various tool motion commands (G-group), machine service commands (M-group), and so on.
4	Command line	A single G-Code file S/W instruction. The contents of a CNC G-Code file is an ordered list of CNC command lines.
5	CNC G-Code Interpreter	The S/W application or program component that interprets G-code command lines and segregates into various groups like motion command G-group, services command M-group, and so on.
6	CNC G-Code Interpolator	The S/W application or program component that converts the G-Code motion command lines into signal commands and save them in a CNC Signals file.
7	Signal command	A single signal file S/W instruction.
8	CNC Signal file	The contents of a CNC Signal file is an ordered list of CNC signal commands. The term reference signal used in most literature is synonymous to this signal command. A reference signal basically refers to the signal after CNC interpretation and interpolation of a single G-Code line command (G-group, M-Group, etc.).
9	CNC Signal Driver	A S/W application or program component that reads the CNC Signal file and interprets a signal command. This application then generates the appropriate H/W electrical pulses using a specific hardware device. The pulses are finally sent to its destination, for example, the servo-driver and servo-motor hardware pair.
10	CNC Conversion from software to hardware electrical pulses	This conversion is the task of the CNC Signal Driver software. It is the boundary or interface point in CNC machine operations, where software codes generate actual physical electrical pulses. The electrical pulses, for example, drive the servo-driver and servo-motor pair, meaning, ultimately driving the CNC machine. The CNC hardware executes only on recognizing electrical pulses. The software codes are not of concern to the CNC machine.

Table 1.1: CNC Terminology Part 1 of 2

CNC Terminology Part 2 of 2		
11	CNC Motion Controller	The S/W component that implements and controls, for example, CNC tool motions, like direction, velocity, position, torque, acceleration, deceleration, feedrate and so on. This is to achieve tool accuracy in following path trajectory, tool motion smoothness, and avoid tool jerks.
12	CNC Error Controller	The S/W component that implements and controls, for example, CNC tool trajectory or path tracking, path error computation, path monitoring and compensation, lookahead control, adaptive control, iterative control, predictive control, AI control, and so on.
13	CNC Services Controller	The S/W component that controls CNC service functions like to start and stop the tool, to start and stop lubrication fluids, to pause and change tool cutters, and so on. It also monitors and controls other auxiliary services like excessive vibrations, over-currents, over speeds, broken tools, over heating and so on.
14	CNC HMI Controller	The Human-Machine Interface (HMI) S/W component that handles inputs/outputs with humans to control the CNC machine. This component also provides display and monitoring services for the entire operation of the CNC machine.
15	CNC Control Loop	<p>The S/W component that coordinates and controls the timely executions of the following software sub-components:</p> <ol style="list-style-type: none"> <li>1. CNC G-Code Interpreter</li> <li>2. CNC G-Code Interpolator</li> <li>3. CNC Signal Driver</li> <li>4. CNC Motion Controller</li> <li>5. CNC Error Controller</li> <li>6. CNC Services Controller</li> </ol> <p>When the CNC Control Loop execution completes and exits, the CNC machine operations is considered completed or execution finished.</p>
15	CNC Controller software (CCS)	The complete S/W application that combines the CNC Human-Machine Interface (HMI) Controller with the CNC Control Loop and its associated sub-components.

Table 1.2: CNC Terminology Part 2 of 2

## 1.6 CNC System Hardware and Software

### 1.6.1 CNC Hardware Components

The hardware for the CNC machine varies from low-end machines to very high end commercial systems. We list below some basic hardware for a minimal CNC system.

1. electrical signal generator device
2. motor-controller and its electric-motor pair
3. computer (normal desktop, laptop or dedicated server)
4. display and monitoring device
5. devices for inputs/outputs and machine tools
6. framework assembly for the complete CNC machine
7. other machine accessories (power supply, cooling system, etc)

### 1.6.2 CNC Software Components

Similarly, the software for CNC machine varies among different machines, from semi-automatic to fully automatic control. We list below some basic software requirements for a minimal CNC system.

1. Computer operating system (RTOS or GPOS)
2. CNC Control Software (CCS) comprises
  - CNC Control Loop
  - CNC Human-Machine Interface
3. CNC Control Loop comprises
  - CNC Interpreter
  - CNC Interpolator
  - CNC Motion Controller
  - CNC Error Controller
  - CNC Signal Driver
  - CNC Service Controller
4. Depending on CNC machine usage, software language compilers for the OS platform may be required for development, especially when the user is a systems programmer that wishes to perform customization.
5. Other associated software libraries with API may be required for external interfacing with the CNC Control Software (CCS).

### 1.6.3 CNC Interpreter

#### 1.6.3.1 Primary memory data storage

In conventional practice, the task of the interpreter is to read a G-Code file, interprets the command line blocks in the file, and stores the interpreted data in memory (we call it primary) for use by the interpolator. When storing activity for one command line block completes, the interpreter reads and interprets the next command line block. In the duration of interpretation of the next command line block, the CNC machine motors may still be executing machine moves based on the previous command or have completed the moves. The expected event is that the moves along the different axis-of-motions for the previous command have completed.

#### 1.6.3.2 Slow interpretation execution

Consider the case of a single processor computer. If the time to interpret the command block is longer than the time to finish the previous command line execution (meaning slow interpretation), the running motors of the CNC machine will have to pause momentarily, because the motors are waiting for completion of interpretation of the next command block. This stop happens because there is only one CPU processor, and that processor is still occupied with interpretation of the next command.

The solution for preventing the machine from regularly stopping (jerking) is to create an internal buffer that temporarily stores the interpreted data. The buffer must always keep a sufficient number of interpreted data for use by the interpolator. This is the usual practice.

For a multi-processor or multi-core computer system, the interpretation of the G-code command line blocks can be executed in parallel using dedicated cores. For example, with a 4-core processor, 3 cores can be used for interpretation while the 4th core is used for CNC loop control. However, it is very important that the organization of interpreted data saved to memory preserves the exact command sequence of the originating G-code file.

One solution is to create a special data structure in memory that accomplishes this ordering. This data organization structure is not necessarily sequential or circular. In addition, with large, fast random access memories, and fast processor speeds available today, for fast access times of interpreted data in memory, we may consider more efficient storage methods, for example, graph or tree data structures.

#### 1.6.3.3 Efficient data I/O methods

The first task of the interpreter is to read the G-Code file, conventionally this happens in sequential order. Reading a file from external storage is very slow compared to reading from computer memory. With parallel I/O for file operations using Message Passing Interface (MPI) library (available since 1997), we may consider executing multiple processes to read and write to a file in parallel.

This means that we can pre-process the G-Code file into a format suitable for parallel file I/O. Reading the G-Code file will be a lot faster. If the G-Code file is very large, using solid state disks (SSDs) to store the G-Code file is another possible option.

## 1.6.4 CNC Interpolator

### 1.6.4.1 Secondary memory or file data storage

G-Code interpolation is the next task after G-Code interpretation. Similarly, in conventional practice, the interpolator sequentially reads the interpreted data from data buffer in primary memory (output of interpretation), calculates the position and velocity for each axis, and stores the result in a secondary memory FIFO (first in first out) buffer for later use by the CNC Signal Driver software controller.

The interpolation results are simply coded command signals that drive the CNC motors or other CNC service functions. If interpolation results are saved in a physical file (persistent), the file is named the CNC Signal file. We will show some of our own designs of CNC signal files that we have used in our previous research work,

The main purpose of saving (capturing) the CNC interpolation results into a physical file (CNC Signal file) as secondary memory instead of real memory buffer (non-persistent) on the computer, is for record keeping, inspection and investigation, such that the same CNC run configuration can be re-executed or played-back at another time.

In online conventional mode, interpretation and interpolation tasks are computed, results stored in memory (not in a file) and then directly fed into the CNC control loop while the loop is running. Whereas, in offline mode, the interpretation and interpolation are pre-computed, results stored in a file (CNC Signal file) and later fed to the CNC control loop.

The term online mode here means, as each command line block interpolation is completed, the resulting coded command signal is immediately fed and executed by the CNC control loop. In offline mode, the CNC control loop fetches the coded command signals from the CNC Signal file, line by line for execution.

The advantage of running in offline mode is the ability to analyse all command signals that are to be fed to the CNC control loop in the future. This means by reading the CNC Signal file, we can look-ahead (look forward) to the exact command signals that are going to be fed sequentially to the CNC Control loop. With this information we can calculate ahead of time, for example, the expected contour error for each move, the expected accumulated error after a few moves, the expected tool velocities for future moves, and so on. This analysis allows us to correct for contour tracking errors, for example, using compensation, adaptation and other error control techniques yet to be considered. This really means we can modify command signal instructions for future moves. This is where the CNC Error Controller and CNC Motion Controller software components comes into play.



After reading one command line "interpreted" data from primary memory, the interpolator will know what kind of instruction to perform. For example, the next position to move to from the current position, the velocity to make the move to the next position, the type of geometric move to execute (linear, circular arc, parabolic or splined segments) for that path segment. The next position to move and the type of move to perform is considered as geometric or trajectory path data. With geometric path data, the distance to be covered by the move will be calculated by the interpolator.

#### 1.6.4.2 Interpolator main task

The main task and responsibility of the CNC interpolator is to generate and send pulses to the respective motors at the corresponding axis-of-motions as commanded by the geometric path data to reach its target position. The interpolator must generate the right number of pulses to cover the distance for the move. It also has to generate the correct number of pulses per unit time (pulse frequency or pulse feedrate) to achieve the commanded velocity for the move.

These calculations are intensive and have to be conducted for all of the motors at all of the axis-of-motions. Imagine the computational needs of different CNC machine configurations, like the 3-axis, 5-axis and 9-axis CNC machines. We realize here that having a multi-core computer and implementation of true parallel computing offer their advantages.

Essentially, the job for the interpolator is to make sure that pulses sent to the different axis-of-motions are coordinated, synchronized, and timely, such that the CNC tool accurately traces (tracks) the commanded path data trajectory. In CNC path tracking, a contour or trajectory error in tool motion means the CNC tool movements have deviated from the intended path contour.

#### 1.6.4.3 CNC Accuracy - Basic Length Unit (BLU)

In CNC motion, the distance covered or displacement per pulse determines its accuracy. For example, if an axis can move at the rate of 0.002 mm per pulse, the accuracy of the CNC system is 0.002 mm. Each pulse contributes to a move. This translational displacement per pulse in CNC is called the basic length unit (BLU). In this example, in order to move a linear distance of 20 mm, we need to send 10,000 pulses ( $0.002 \times 10,000$ ). If we want to generate a move with velocity 20 mm per second, we need to generate 10,000 pulses per second or run at 10 KHz pulse frequency.

#### 1.6.4.4 Velocity - Signal Pulse frequency

The pulse frequency limit is a physical constraint in the design of the CNC machine. Consider an example where the maximum pulse frequency of the pulse generator device (hardware) is capable of driving is 50 KHz, and the maximum pulse frequency the servo-driver and servo-motor pair can handle is 30 KHz. In this example, for the combination of pulse generator and servo-motor, the maximum pulse frequency for the CNC machine is only 30 KHz. Even though pulse rates can go up to 50 KHz, the motor can only handle up to 30 KHz. The

bottleneck is the servo-motor.

We know that in software, there is no limit to the value we can set for the driving pulse frequency (number of pulses per second, or feedrate), because it is just a number setting parameter in software, but in hardware there is a real physical limit.

If we use a lead-screw or ball-screw shaft for CNC linear displacement, the finer the screw thread pitch (thread interval distance), the more accurate is our machining. For very fine linear moves, we need very high pulse feedrates to finish the machining job within a reasonable amount of time. To get high feedrates, we need high frequency pulse generators and compatible high frequency servo-servo and servo-motor pairs.

The variable frequency (feedrate) error control method is a common control algorithm in CNC interpolation. This is done by suitably varying the frequency in time so that the CNC tool moves smoothly (smooth velocity profile) as it traces the contour path trajectory. Smooth velocity profiles mean continuous moves at the joints of the path segments. The smooth velocity profile concept is not about contour path displacement error, but it is affected by it (conflicting effects). This conflicting effect is discussed in the next section.

#### 1.6.4.5 Tool Velocity Profile

A simple way to look at conflicting effects of contour path displacement errors against smooth velocity motions is as follows. Consider the analogy of driving a car. In order to quickly reach the end point of a curve on the road (arc segment), we need high velocity moves. With high velocity moves we cannot negotiate the sharp curve quickly because of momentum effects of the car. We will go off track, meaning, we created a contour error. If we drive at a lower velocity (slow), we will trace the contour path very well (good tracking profile). However, it takes a much longer time for us to reach the end point.

Driving a CNC machine is no different from driving a car, in fact, it is similar. Today, we have autonomous and self-driving cars under computer control that can do this job quite (not very) well. More on this will be discussed in the section on literature survey.

The tool velocity profile is concerned with acceleration and deceleration of the CNC cutting tool as it traces the commanded geometric contour path. CAD/CAM systems have to divide curves into a large number of segments while maintaining the set of contour error limits. For example, the segments comprise short linear (G01) segments, circular arc (G02, G03) segments or NURBS (G06 series for surfaces) segments generated by advanced high end CAD/CAM systems. Usually, low end CAD/CAM systems do not generate NURBS G-Code.

Due to the short segments, most of the time spent during interpolation is on stop-start motions (M-Group), typically accelerating, decelerating, or pausing between instructions. The frequent starts and stops for very small linear motions for every segment cause jerks during interpolation, especially at the junctions of the segments.

This results in discontinuous feedrate (velocity) profiles, jerky and generally un-smooth overall machining process.

#### 1.6.4.6 Velocity profile controller

The velocity profile controller is also known as Acceleration/Deceleration (ACC/DEC) controller in older CNC literature.

If position control is executed by using data generated from the interpolator, whenever tool movement starts and stops large mechanical vibrations and shocks occur. In order to prevent mechanical vibration and shock, the filtering for ACC/DEC or velocity control is executed before interpolated data is sent to the position controller. This method is called the ACC/DEC after-interpolation method.

There is also the ACC/DEC before-interpolation method, where velocity control is executed before interpolation. This before-interpolation technique is akin to a look-ahead strategy. We do not consider this strategy as predictive control because the commanded contour path trajectory to be followed is already known for certain. We know for certain the exact position to go as our target. More will be described in the section on literature review.

The data from an velocity controller is sent to a position controller, normally for position adjustment. A position control mechanism typically means a PID controller. This PID controller issues velocity commands to the motor driving system in order to minimize the position difference between the commanded position and the actual position. The actual position is found from the encoder feedback in servo-driven motors.

For example, for our CNC Research machine, the commercial, industry standard and high end Panasonic Minas A4 servo-driver has a built-in configurable electronic circuit PID controller. The servo-driver can be configured to run on any one of three modes: position control, velocity control and torque control. The parameters for the PID controller can be set manually or calibrated automatically against the particular CNC machine characteristics to handle the velocity profiling problems (momentum effects).

Based on using the Minas A4 servo-driver, our experience on PID control at the electric-motor side (receiving end) is very satisfactory. For this reason, in this research project the velocity PID control handling is not our focus and therefore will not be included in our scope. However, we will still consider velocity control but at the CNC Interpolation software level on the trajectory path, and not at the real electric servo-motor end level.

In this project, our focus shall be more on contouring control, that is, making sure that our CNC tool contour tracking is accurate against the G-Code instructions. We are not even concerned with prior errors generated in CAM processing conversion of 2D/3D model drawings to produce G-Codes because we consider the start of our research project as beginning from the provision of G-Codes itself, not earlier.

### 1.6.5 CNC Signal Driver software

The CNC Signal Driver is the software component that comprises an important part of the CNC Control Loop. This software program reads the CNC Signal file and interprets a signal command. It then generates the appropriate electrical pulses (hardware) using a specific hardware device. The pulses are finally sent to its destination, for example, the servo-driver and servo-motor hardware pair. The CNC Signals file allows for offline CNC execution.

The Signal Driver software is simply tasked with converting software commands into electrical pulses. This conversion is the boundary or interface point in CNC machine operations, where software codes generate actual electrical pulses. The electrical pulses, for example, drive the servo-driver and servo-motor pair, meaning, ultimately driving the CNC machine. The CNC hardware executes only on electrical pulses. The software commands (instructions) are not of concern to the CNC machine.

Note that the CNC Signal Driver software itself cannot directly generate electrical pulses. It has to fetch from the CNC Signals file, the signal codes and send the software codes to a real physical device (hardware), called the CNC Device Driver. It is this actual hardware device that generates electrical pulses. We explain this further in the next section.

Even though anyone can buy the signal/pulse generator hardware device from the market, the generation of software signals and its contents are proprietary. This is another strong motivation, why in this research project we want to develop our own CNC Interpreter and CNC Interpolation software.

In addition, despite having the same signal/pulse generator hardware device, everyone will have their own way of creating and defining the contents of CNC Signal file. We have experienced the creation and definition of our own signals file and have executed them successfully. The signals file is machine dependent because it is specific for each pulse generator hardware. We will show this in the section on previous project experiences.

The CNC Signal file is useful in the following sense. In offline CNC operation mode, both interpretation and interpolation are pre-computed, results stored in a file (CNC Signal file)

and later the contents of the file fed to the CNC control loop.

Whereas in online CNC operation mode, both interpretation and interpolation tasks are computed in memory and then fed immediately and directly as it arrives into the CNC control loop while the loop is running. The results of computations for the software signals are held in memory and will be lost when the computer shuts down.

However, with multi-core computers like 8-cores, we can have 1 of the cores dedicated to just saving the results as it arrives. that is, by reading from memory and writing to a file like a "memory dump" for persistent storage.

This provides us with another opportunity in our research project to implement parallel computations. One processor core assigned solely for signal dump task while the other cores are assigned for normal tasks like driving the many axes of the CNC machine, processing of interrupts from the CNC Services controller, and so on.

### 1.6.6 CNC Device Driver hardware

The CNC Device Driver is the hardware device that interfaces between the CNC software and the CNC hardware. This is the boundary point where software commands get converted to electrical signals. This device receives software commands and generates electrical signals (either digital or analog signals) and transmits the signals to the motor control driver.

This hardware device could be an internal device like the computer parallel port, USB port, or serial port, or a specialized embedded board (ISA, PCI, PCI Express) like motion control boards installed inside the computer or an industry external interface board interfaced with the computer. There are many motion control board offerings available in the market for this specialized category of hardware.

Depending on configuration and type of signal generator device, communications can be unidirectional or bidirectional between the CNC control computer and CNC hardware. The actual signal generator device does not matter. The CNC machine operations will work as long as the device can generate appropriate electrical signal pulses to drive the CNC motors.

#### 1.6.6.1 Research pulse generator devices

On our CNC research machine, the signal generator devices that we have used and tested successfully (from 2010-2018) comprise the following:

1. Personal Computer (PC) parallel port, ref[?], links[??], [??], [??].
2. Arduino Due board, ref[?], link[??].
3. Heber X10i USB board, ref[?], link[??].
4. Raspberry Pi 2 and Raspberry Pi 3 SBC boards, ref[?], links[??], [??].
5. Banana Pi SBC board, ref[?], link[??].
6. Velleman K8000 Parallel and K8055 USB interface boards, ref[?], links[??], [??].
7. BeagleBoard xM SBC board, ref[?], link[??]
8. Prolific PL2303 USB and PL2305 Parallel ports, ref[?], link[??].
9. Digilent Nexys-3 Spartan-6 FPGA development board, ref[?], link[??].

As we mentioned earlier, the creation and definition of the contents of CNC Signal file for the different pulse generating hardware devices listed above are different from each other. The signals file is machine dependent because, for example, the pin assignments, the registers and access APIs are different.

Details of CNC device driver implementations in our research using the above devices as signal generators are provided in the section on related research work.

### 1.6.7 CNC Motor Control Driver hardware

In industrial practice, most software controlled electric motors comes as a pair, consisting of the motor-control driver and the actual electrical motor itself. For example, for a CNC servo system, it consists of a separate servo-driver and a servo-motor hardware pair. In some systems, the motor-control driver is built as a part attached to the electric motor.

The motor-control driver is a hardware device with built-in electronics that commands and controls the actual running of the electric motor. This hardware, on one side is wired to the CNC driver device like the parallel port on the computer, and on the other side is wired to the CNC electric motor. In our CNC Research machine, this motor-control driver is the Panasonic Minas A4 AC servo-driver hardware.

The motor-control driver hardware usually contains embedded software that performs control functions like PI, PID, and encoder feedback. The control parameters can be configured manually or by external software. The motor-control configuration parameters depend on the choice of driving schemes, for example, position control, velocity control, torque control, open-loop and closed-loop controls. Some motor-control hardware operate in open-loop control, for example, stepper-motor control without feedback, while others are used for close-loop control, like servo-motor control with feedback.

### 1.6.8 CNC Electric Motor hardware

There are many different types of electric motors used in industry, some are manual hardware controlled while others are computer software controlled. For software controlled motors, they come in a pair, the motor control-driver and the electric-motor pair.

In the CNC industry, an electric motor can be direct current (DC) or alternating current(AC) driven. Each type has its own advantages and disadvantages. Some motors are used for open-loop control while others are used for close-loop control.

### 1.6.9 CNC Control Loop design

The actual control loop design varies between different kinds of CNC machines. However, they share the same concepts and principles, that is, to run the system correctly obeying their designed parameters and limits.

The following are some control methods: open-loop control, closed-loop control, position control, velocity control, torque control, iterative control, adaptive control, compensation control, predictive control, artificial intelligence control, optimal control, security control, safety control, and so on. These control methods or strategies will be addressed in the section on literature review.

#### 1.6.9.1 Control objectives

In general, the first objective of control is to ensure that the system performs its specified function. The second control objective is to ensure that any variations in performance must

stay within its prescribed and designed limits. A successful control design must satisfy these two objectives.

A control system consisting of interconnected components is designed to achieve a desired purpose. The standard software, engineering practice today, is to build systems with modular component architecture. In modular designs, new components can be added and existing components can be modified or removed without affecting the overall behaviour of the system.

### 1.6.9.2 Control loop operations

For example, the CNC software control loop operates by invoking and executing separate functional components according to some control algorithm. Depending on the design of the control algorithm, the invoked components may run sequentially, concurrently or in a parallel manner. (parallelly is acceptable in old English usage ... ha ha ha).

The control loop is considered the sole director that is tasked to orchestrate the interaction and interfacing among the different component functions within the system, and with the external environment. Thus, the construction of the control loop algorithm is the most important software component in the system. It is called a loop because it is supposed to run in-resident, stay continuously active, waiting to service any command request as it comes, and act to those events accordingly. The control loop is a service or daemon type of running program. If the control loop program dies, the entire system dies.

### 1.6.9.3 Control concepts

Control algorithms are designed using to the following concepts (PRMCC).

1. a plan - the step-by-step sequence of actions to follow
2. a review - the action of monitoring some parameters, act accordingly if the values are out of range
3. a measure - the action of taking measurement of some parameters, used for progress checking and decision making
4. a coordination - the action of timely communicating with other components inside and outside the system
5. a control action - the decision on what action to take given the current state of the system.

### 1.6.9.4 Programming principles in control

In software design for control algorithms, the following are a few programming principles that apply:

1. process-driven programming - this is a step-by-step execution of functions, pre-planned and laid out to achieve some defined objectives
2. interrupt-driven programming - this is a request or notification event to the control director that an event occurred which required action



3. target-driven programming - this is a specification contract that the software must achieve the target by whatever means necessary
4. optimal-control programming - this is a specification contract such that software parameters stay within certain agreed limits by whatever means necessary

### 1.6.9.5 Issues in control loop algorithm

For the CNC control loop algorithm, all of the above mentioned concepts and principles apply. The design of the algorithm is proprietary. No sensible manufacturer will reveal this secret. In this research, we will build our own CNC control loop algorithm.

For the CNC control loop in particular, for process-control the control algorithm must handle machine services, like start and stop the motors, monitor tool locations, monitor path tracking errors, monitor velocity limits, etc, in order to achieve process targets and act accordingly.

For safety-control, the control algorithm must monitor temperature limits, friction limits, heating limits, vibrations, limits, etc, in order to service interrupt events and act accordingly.

For optimal control, the control algorithm must act by invoking and executing appropriate functions accordingly in order to achieve the optimal value (target maximum or minimum) of some dependent variable based on current values of independent variables or parameters.

Essentially, the control actions in the CNC control loop program work in a similar manner to a human being who has to make simultaneous control decisions and acts accordingly when faced with many things to do at one time. The control sensors for a human being is akin to the monitored parameters in the CNC machine.

The CNC control loop simultaneous multiple objectives are, as examples: to minimize contour error in tool path tracking, to ensure tool motion smoothness, to ensure the machining job is completed within reasonable time, to ensure minimum vibrations, to ensure heating is within tolerance, and so on, but not to forget that the machining job must be executed safely.

As a consequence, the design of the CNC control loop is not an easy task. It is one of the challenges that we will undertake in this research project. We will discuss more on this in the section on literature review.

### 1.6.10 Commercial CNC Control Software

Based on the best CNC control software survey in 2017 published on the internet, [?], the following are some results for large industrial CNC machines, and small and medium shop-hobbyist type CNC systems.

### 1.6.10.1 High end CNC machines

For large and commercial CNC machines, the CNC control software are built special-purpose for the respective machines. Some famous brand names are as follows:

1. Fanuc, [?].
2. Haas, [?].
3. Mazak, [?].
4. Siemens, [?].
5. Centroid, [?].
6. Heidenhain, [?].
7. Mitsubishi, [?].
8. Allen-Bradley [?].

### 1.6.10.2 Low end CNC Machines

For small and medium size commercial CNC machines, the CNC control software are similarly built special-purpose for the respective machines. Some famous brand names are as follows:

1. Mach CNC, [?].
2. PathPilot/LinuxCNC, [?].
3. Probotix CNC, [?].
4. Planet CNC, [?].
5. Eding CNC, [?]
6. UCCNC Motion Control. [?].

### 1.6.10.3 Survey results

The survey concluded that FANUC and HAAS are way ahead of everyone else at the high end market, while Mach3 and PathPilot-LinuxCNC rule the low end market. In this proposed research, we will be dealing only with LinuxCNC because it is cost-free but PathPilot is not.

## 1.7 Research Motivations

### Commercial interest

Recall that CNC interpolation is the task that generates the actual reference commands that drives the CNC tool along the different axis-of-motions in the machine, such that, it accurately follows the desired machining path, in a timely and coordinated manner.

On this fact, it can be said that interpolation is both the "brain and heart" of the CNC machine. The brain refers to the logic (program or algorithm) that gets the CNC running correctly and accurately, while the heart refers to the continuous, timely and coordinated supply of just the right power (pumping blood, oxygen and nutrients) to the CNC machine. Many people can copy, duplicate or reverse-engineer the physical aspects of the CNC machine because they are visible and physical, whereas the CNC interpolation software is intangible, and not easy to copy if properly protected.

It was said that you can buy the machine, and the completed compiled software, but you cannot buy the source code that shows the internals on how the software was developed, so that you can learn from it. You need to develop one on your own and then learn from it.

### Ownership and accomplishment

We do not have our own CNC interpolator because existing interpolators are proprietary, meaning, owned by the CNC manufacturers and developed for their own unique commercial machines. It is plain common sense that manufacturers will not share or reveal their commercial secrets, except possibly marketing and generally well-known information.

We experienced real situations where suppliers interrogate us, the seller is suspicious of us, for example, regarding the purpose of our purchase when comes to high intellectual property products. Suppliers want to ensure that their products do not land into the wrong hands, particularly potential future competitors. They want to sell only to users but not knowledgeable developers. That is the game. The only way out is to always stay ahead, or always be the faster man than the fastest man.

Therefore, it is of utmost interest for us to develop our own CNC interpolator. We know that in Malaysia, statistics show that more than 80 % of commercial CNC machines and the like in use are imported.

### Entrepreneurship

We are interested and excited to manufacture our own commercial CNC machine in the near future, getting into the competitive CNC machining market and initially targeting at specific segments in our local market. In a borderless world today, we can even go to internet marketing, as many small and medium enterprises do today. We are excited means happy in the heart, for example, to study, explore, learn, undertake, execute and discover new things.

Our CNC product, interpolator plus machine, must be run effectively and efficiently. We must be effective means it must take effect, for example, to do exactly and correctly what it is supposed to do, that is according to its specifications. We must be efficient means be minimal in computation time and resources, for example, executes fast, speedy, compact and does not consume large resources.

### **Technical challenges**

It is challenging technically, for example, to study the look-ahead control and feedrate filtering issues in CNC machining and come up with our way of addressing those issues. Our success in providing alternative solutions to problems will arouse a personal sense of accomplishment, satisfaction combined with a feeling of worthiness, one who is useful and can contribute in some small way for the benefit of mankind.

### **Future possibilities**

With the exception of *non-computable functions* in computing, software programming and control opens unlimited possibilities in what software technologies can do and accomplish.

The prospects extend from directing sequential step-by-step commands, parallel executions, sorting, search, scheduling, exotic software concepts like futures and asynchronous executions, communication channels, and so on, are all recent ideas coming from the human mind to automated logical reasoning, software self-healing, fuzzy reasoning, self-learning, and so on, conducted within and by the software itself.

It is not easy to think of doing things in parallel, executing many things in overlapping time, not necessarily concurrent, but that is reality of nature. It will be extremely satisfying, not only being able to run things in parallel, but doing so in true realtime elegantly, while meeting both the start and end design deadlines. The thoughts to come up with software solutions addressing those issues are truly exhilarating.

## 1.8 Scope of Research

The scope of work proposed for this research study are as follows:

1. Start from the provision of G-Codes, specifically RS274D NGC standard G-Code.
2. Implement reference-pulse CNC interpolation for computational efficiency.
3. Conduct look-ahead and feedrate error compensation in G-Code interpolation.
4. Execute realtime, parallel, online/offline computations for the CNC control loop.
5. Compare appropriate parallel execution methods for 2D/3D G-Code interpolation
6. Address designs for extension to 3-axis and 5-axis interpolation implementations.

## 1.9 Proposed Research Title

The proposed research title shall be "A realtime and parallel look-ahead control and feedrate compensation strategy for CNC reference-pulse interpolation."

## 1.10 Expected knowledge contributions

The expected knowledge contributions for this research study are as follows:

1. an implementation of a simple, practical and achievable strategy in CNC interpolation
2. a technique that utilizes both realtime and parallel execution in CNC machining
3. a contribution in innovative ideas for an efficient design of the CNC interpolator

## 1.11 Summary on Introduction

In this introductory chapter on our proposed research project, we began with a short overview of CNC, followed by the extensive adoption of software techniques in the control of CNC machines, particularly, contour tracking control executed in the CNC interpolation loop.

We briefly described recent designs in CNC machining environments, covering new cutting tools and technologies, and innovative software algorithms that address issues of path motion smoothness, machining speed control, and path error reduction.

The typical hardware and software requirements for a CNC machine were covered. A typical functional process flow for CNC machine operation was described. We illustrated what realtime, parallel and concurrent execution concepts mean to the common person compared to the correct understanding in software terminology through clear examples.

We briefly explained the functions of various software components in a typical CNC system, the G-code interpreter, the G-Code interpolator, the motion controller, the error controller, the signal driver, service controller, the human-machine controller and the overall CNC controller loop.

The interface and boundary point between software and hardware in CNC machine was specially defined, by differentiating the terms signal driver software against signal driver hardware. The signal driver hardware is generally referred to as electrical pulse generating devices.

Some signal driver hardware interface boards that have been used in our previous research projects were listed. In addition, the issues that need to be addressed by each of the software components to work with those signal driver devices were also discussed.

On the hardware side of the CNC machine, we described the industry practice of a compatible set of motor-driver and electric-motor pair, such that there is no mismatch of performance, for example, pulse frequency handling. In standard practice, the software control loop of the CNC machine only interacts with the motor-driver hardware, not directly to the electric-motor. We shared survey results on commercial companies with leading products in the CNC machining market, and identified the major brands that offer high end and low end CNC machines.

Finally, we described our motivations for conducting research on CNC machines, our proposed research topic, our proposed the scope of research and the expected contributions of the work.

In summary, the focus of our research is CNC interpolation. We restate, that CNC interpolation is about the task of generating and sending signal pulses to the respective electric motors at the machine's axis-of-motions, in a coordinated and timely manner, such that the machine tool accurately tracks the desired contour path in the move from the current position to the next position. To achieve this task, the interpolator must generate the right number of signal pulses that achieves the desired distance for the move, and the correct pulse feedrate (frequency) that achieves the desired velocity for the move.

## 2 Literature Survey

Semi-systematic review with a table of comparisons. In the appendix. Only summary here.

### 2.1 Reviews - CNC State of the Art

[?] CNC Algorithms for Precision Machining: State of the Art Review

As geometry of machined parts becomes complex the demands for more precise and faster machining using advanced computerized numerical control (CNC) are increased. Especially, recently improved computing power of CNC enables the implementation of the complicated control algorithms. Consequently a variety of intelligent control algorithms have been studied and implemented in CNC. This paper reviews the 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 are discussed. Regarding contour control algorithms recently developed multi-axis contour control methods are reviewed.

[?] Recent development in CNC machining of freeform surfaces: A state-of-the-art review

[?] Interpolator for a CNC System

[?] Design of Computer Control for Manufacturing Systems

[?] - Freeform surfaces, also called sculptured surfaces, have been widely used in various engineering applications. Freeform surfaces are primarily manufactured by CNC machining, especially 5-axis CNC machining. Various methodologies and computer tools have been developed in the past to improve efficiency and quality of freeform surface machining. This paper aims at providing a state-of-the-art review on recent research development in CNC machining of freeform surfaces. This review primarily focuses on three aspects in freeform surface machining: tool path generation, tool orientation identification, and tool geometry selection. For each aspect, first concepts, requirements and fundamental research methods are briefly introduced. The major research methodologies developed in the past decade in each aspect are presented with details. Problems and future research directions are also discussed.

### 2.2 Overview of current issues in CNC

TO DO

### 2.3 G-Codes Standards

### 2.3.1 NGC RS274D G-Codes

TO DO -

G-code (also RS-274 NGC), which has many variants, is the common name for the most widely used numerical control (NC) programming language. It is used mainly in computer-aided manufacturing to control automated machine tools.

ISO 6983-1:2009 specifies requirements and makes recommendations for a data format for positioning, line motion and contouring control systems used in the numerical control of machines. ISO 6983-1:2009 helps the co-ordination of system design in order to minimize the variety of program manuscripts required, to promote uniformity of programming techniques, and to foster interchangeability of input programs between numerically controlled machines of the same classification by type, process, function, size and accuracy. It is intended that simple numerically controlled machines be programmed using a simple format, which is systematically extensible for more complex machines.

### 2.3.2 NURBS G-Codes

Low end CAD/CAM systems do not generate NURBS G-Code.

Non-Uniform Rational B-Splines NURBS have been used by CAD systems for some time. That is why it seems so natural that CNCs should be able to employ tool paths that are also defined in terms of NURBS. However, most CNCs today instead require contoured tool paths to be defined using straight lines, or chords.

And this long-practiced approach can lead to inefficiencies familiar to almost any die or mold shop. Using chords to define complex geometries accurately results in large, data-dense program files that historically have been difficult to manage and slow to execute. The development of NURBS-interpolating CNCs promised programs that could define the same complex geometries with fewer blocks of code, and thus could provide some relief for the data-flow bottlenecks.

But something happened along the way. CNCs became more powerful, and powerful CNCs became less costly. Compared to the best controls available only a short time ago, many new controls today offer superior networking capability, cheap memory and faster processing speed. These improvements mean enormous program files for complex workpieces are easier to deliver to the CNC, where they can be stored entirely instead of drip-fed. And once there, the programs can be executed more efficiently. Processing rates on the order of 1,000 blocks per second, combined with CNC look-ahead features to smooth out inertial effects, can let a new control today execute even a long series of very tiny chords fast enough to keep a machine moving accurately at a high feed rate.

Using nurbs interpolation requires not just a CNC capable of it, but also a CAM system able to output nurbs tool paths. And these CAM systems can use a variety of approximations to get from one set of nurbs to the other. Here are just two reasons why:

Surfaces are not tool paths. Mr. Arnone states it this way: "When a plane is intersected with a nurbs surface to create a tool path, a nurbs curve does not result."

In other words, the software is still approximating to get to the tool path. And like any approximation, this one is governed by an accuracy band, comparable to chordal tolerance.

A straight line may be the route between two curves. Some CAM systems translate the CAD geometry into nurbs toolpaths by generating a series of chords first, then translating the chord tool paths into nurbs. For CAM systems that use this approach, there literally is a chordal tolerance at work.



Finally, it's worth noting that at the level where the cutting tool hits the workpiece, the movement is still in straight lines. The CAD model may be represented by nurbs, the CNC may read tool paths in terms of nurbs, but when the CNC communicates movement commands to the processor controlling a given axis, it does this by specifying a target point. That is, a target toward which the axis advances in a straight line.

===== Optimized NURBS Based G-Code Part Program for High-Speed CNC Machining

August 2014 DOI: 10.1115/DETC2014-34884

Conference: ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference

### 2.3.3 STEP-NC G-Codes

[?] STEP-Compliant CAD/CNC Systems for Feature-Oriented Machining

[?] - In the projection toward the development of next generation CNC system, the major problem of current International Standards Organization (ISO) data interface model (ISO 6983) limitations and commercial CNC unit vendor specifications dependency were found in the CNC machines and systems. Later on, an ISO standard known as Standard for The Exchange of Product Data (STEP) or ISO 10303 was introduced to provide remedy for the problems of current data interface model limitations in Computer Aided Design (CAD)/Computer Aided Manufacturing (CAM) systems. After that successful implementation, the standard further extended to implement the STEP features on the CNC, for that a new standard known as STEP-Numeric Control (NC) or ISO 14649 was introduced. This standard has the abilities to achieve the aims of modern CNC systems. However, for enabling these facilities into the CNC machine, the machines need to be independent from any of the vendor specifications. Open Architecture Control (OAC) technology provides a more open environment to the CNC machine. In the race towards the development of next generation CNC, the combination of the STEP-NC and OAC technology became the hot topic of the research. However, in this work both ISO data interface model interpretation, its verification and execution has been highlighted with the introduction of the new virtual component technology based techniques. The system was composed of ISO data interpretation, 3D simulation and machine motion control modules. The system was also tested experimentally and found to be very satisfactory.

The development of STEP started in 1984 as a successor of IGES, but due to the complexity of the project, the initial standard was only published in 1994. Initial Graphics Exchange Specification (IGES) is a neutral file format designed to transfer 2D and 3D drawing data between dissimilar CAD systems. The IGES standard defines two file formats: fixed-length ASCII, which stores information in 80-character records, and compressed ASCII.

## 2.4 Reference-Pulse interpolation

In the Reference-Pulse interpolation method, reference signals from the computer are transmitted as a sequence of reference pulses, basically a sequence of external interrupts sent to the control loop of the CNC machine drive. In this interpolation method, the computer

transmits a sequence of reference pulses to each axis-of-motion, where each pulse produces one BLU (Basic Length Unit) of movement.

Each axis-of motion is controlled by two pulsed lines, one for clockwise rotation (CW) and the other for counter-clockwise (CCW) rotation. The accumulated number of pulses for each axis represents the location of the tool on that axis. The pulse frequency, generated by varying the time spacing between pulses, is proportional to the velocity of the tool along the axis. This frequency represents the axis feedrate.

The reference-pulse method can be used for actuating stepper motors in an open-loop system or sent as reference signals to servo motors in a closed-loop control system. For example, a pulse train of varying frequency is output to the servo control module. For each pulse, the servo system for an axis causes an incremental movement along the axis.

All reference-pulse interpolations are iterative and is usually controlled by an adjustable interrupt clock. A single iteration of the routine is executed at each interrupt, and this produces the output pulse. For the computation, the maximum feedrate (feed pulses) generated is limited by the maximum attainable computer interrupt rate. The interrupt rate in turn depends on the computation time of the algorithm.

The faster the computation, the higher will be the feedrate. The number of iterations required to move the CNC machine to a certain distance on a contour path segment for each axis-of-motion is also calculated by the interpolation algorithm.

[?] CNC Machining Technology [?] CNC Controllers and Programming Techniques  
[?] Reference-Pulse circular interpolators for CNC Systems  
[?] - A Reference-Pulse Generator for Motion Control System

## 2.5 Reference-Word interpolation

In the Reference-Word (Sampled-Data) interpolation method, the interpolator runs in an online iterative mode and generate words (sampled data) which are supplied as references to the running CNC control loops. It is a point-to-point control method used to move to the desired position between two successive interpolated points along each axis. The coordinate points to reach from the present position are computed for each iteration and the calculated data is transmitted to each axis in the form of sampled data.

For example in 2D motion, this calculated sample data consists of the next x and y coordinates and their velocities along each of the x and y axes. In a 2D contouring system the tool is cutting while the machine axes are moving. The contour path of the part is determined by the ratio between the velocities along the two axes. During a closed loop execution, the interpolation subroutines continuously provide destinations and velocity set points to the stepper or servo drive systems.

In contrast, in reference-word interpolation the maximum feedrate (velocity) is not limited by the execution speed of the processor. It is the stepper or servo system dynamics that limits the speed rather than the interpolator loop execution time.

[?] Design Parameters for Sampled-Data Drives for CNC Machine Tools  
 [?] Reference-Word circular interpolators for CNC systems  
 [?] Research and Development of Sampled-data Interpolation Algorithm Software in CNC System Based on the Visual C++

## 2.6 NURBS CNC Interpolations

Some of the newer CNCs have NURBs interpolation available, but most controls won't have it unless you order it as an option. NURBs interpolation would let you write a G-code program with the NURBs data, which can make a single pass on a complex surface with just one block of G-code data. It reduces the size of the G-code file, and it also increases the maximum feedrate you can achieve.

FANUC USA uses G06.1 G-Code.

===== G05 P10000 High-precision contour control (HPCC) M Uses a deep look-ahead buffer and simulation processing to provide better axis movement acceleration and deceleration during contour milling

G05.1 Q1. AI Advanced Preview Control M Uses a deep look-ahead buffer and simulation processing to provide better axis movement acceleration and deceleration during contour milling

G06.1 Non-uniform rational B-spline (NURBS) Machining M Activates Non-Uniform Rational B Spline for complex curve and waveform machining (this code is confirmed in Mazatrol 640M ISO Programming)

===== ed (I just got the DXF), but you might be able to apply that reasoning to surfacing. And if you can create a CAD model, CAM it from there.

Real-time NURBS curve interpolator for 5-Axis CNC machining

February 2018

DOI: 10.7736/KSPE.2018.35.2.129

Sungchul Jee

[?] - The first appearance of NURBS is in K. Vesprille's PhD thesis [151], where he makes heavy use of homogeneous coordinates, an approach that goes back to S. Coons and R. Forrest. Riesenfeld [130] realized early that using homogenous coordinates mandates working with projective geometry. This book follows these ideas and attempts to base the theory of NURBS firmly in projective geometry. This way, we gain more insight into theoretical issues as well as practical ones. After a general outline of projective geometry, we introduce conics through a classical projective definition. Using the concept of cross ratios, we arrive at the de Casteljau algorithm for conics. Then we cover areas such as rational Bezier curves curves, NURBS curves and surfaces, triangular patches, Gregory patches, and more. The book closes with some practical examples, including a discussion of the IGES NURBS data specifications.

[?] - Parametric interpolator versus linear interpolator for precision surface machining

[?] - Development of a NURBS Curve Interpolator with Look-ahead Control and Feedrate Filtering for CNC System

[?] A real-time scheme of cubic parametric curve interpolations for CNC systems

[?] Feedrate fluctuation compensating NURBS interpolator for CNC machining

[?] - Accuracy and Error Compensation of CNC Machining Systems

[?] - Computer Numerical Controlled System with NURBS Interpolator

[?] - Development of Real-time Look-Ahead Algorithm for NURBS Interpolator with Consideration of Servo Dynamics

[?] Development of a dynamics-based NURBS interpolator with real-time look-ahead algorithm

The International Journal of Advanced Manufacturing Technology

April 2008, Volume 36, Issue 9 –10, pp 927 – 935 Cite as The design of a NURBS pre-interpolator for five-axis machining

Authors Authors and affiliations

Wei LiEmail authorYadong LiuKazuo YamazakiMakoto FujisimaMasahiko Mori

=====  
<https://www.wittystore.com/g-code-commands>

=====  
 ===== G06.1

Non-uniform rational B-spline(NURBS) Machining M

Activates Non-Uniform Rational B Spline for complex curve and waveform machining (this code is confirmed in Mazatrol 640M ISO Programming)

=====  
 ===== List  
 of G-Codes commonly found on FANUC and similarly designed controls Published In:  
 Tech Explained Hits: 800

G-codes, also called preparatory codes, are any word in a CNC program that begins with the letter G. Generally it is a code telling the machine tool what type of action to perform, such as:

Rapid movement (transport the tool as quickly as possible in between cuts) Controlled feed in a straight line or arc Series of controlled feed movements that would result in a hole being bored, a workpiece cut (routed) to a specific dimension, or a profile (contour) shape added to the edge of a workpiece Set tool information such as offset Switch coordinate systems

=====  
 ===== Warning:

G5.2, G5.3 is experimental and not fully tested.

~~G5.2 is for opening the data block defining a NURBS and G5.3 for closing the data block.~~  
 block 2. Repeat lines between these two codes the end five control points are defined with both their related weights (P) and the parameter (L) which determines the order of the

## 2.7 Contouring control and Error Compensation

- [?] The Error Analysis of Surface Machining on the CNC Machine Tools
- [?] Accuracy improvement of three-axis CNC machining centers by quasi-static error compensation
- [?] Adaptive Error Control Algorithm for High Speed Two-Axis CNC Contouring
- [?] Reduction of the Contouring Error in High-Feed-Speed Machining by Real-Time Tracking-Error Compensation
- [?] Integrated Geometric Error Compensation of Machining Processes on CNC Machine Tool
- [?] Embedded Iterative Learning Contouring Controller Based on Precise Estimation of Contour Error for CNC Machine Tools
- [?] Tracking error reduction in CNC machining by reshaping the kinematic trajectory
- [?] Research on Error Compensation Technology for CNC Machining
- [?] Contour error and control algorithm in CNC machining tool
- [?] Dynamic evaluation of spatial CNC contouring accuracy
- [?] Accuracy and Error Compensation of CNC Machining Systems
- [?] Tracking error reduction in CNC machining by reshaping the kinematic trajectory
- [?] Integrated Geometric Error Compensation of Machining Processes on CNC Machine Tool
- [?] Develop on feed-forward real time compensation control system for movement error in CNC machining
- [?] CNC machining accuracy enhancement by tool path compensation method

## 2.8 Look-ahead control

Conventional CNC systems only provide line (G01) and circular (G02, G03) interpolations, so the CAD/CAM systems have to divide the curves into a large number of small linear and circular arc segments while maintaining the set of contour error limits.

Due to the short segments, most of the time spent in interpolation is on stop-start motions, typically accelerating, decelerating, or pausing between instructions. The frequent starts and stops for very small linear motions for every segment causes jerks during interpolation, especially at the junctions of the segments. This results in discontinuous feedrate (velocity) profiles, jerky and unsmooth overall machining process.

The feedrate look-ahead control algorithm is introduced to deal with sudden changes of feedrate [13-15,23-25]. Look-ahead control is a pre-processing task of the path contour before the real machining. The main goal is to achieve a smooth feedrate profile by looking at deceleration regions in the path.

The feedrate look-ahead control algorithm has three tasks: (1) to detect the deceleration point timely before reaching a linking corner or terminal point, (2) to smoothen and re-plan the feedrate in the deceleration region, and (3) to make the deceleration (braking) meet the capabilities of the physical machine like electrical motors. Braking is very important especially for high speed machining because exceeding the braking capability of the machine can cause unwanted machine vibrations.

The look-ahead interpolation algorithm is executed such that the processing error lies

within a limited range. The look-ahead control strategy can be performed as either an off-line prediction, or as an on-line computing operation [24].

Many methods have been proposed for look-ahead control strategies. Some examples are:

1. Generate a recursive trajectory to estimate and determine the deceleration stage according to the distance left to travel on the segment.
2. Apply a hybrid digital convolution technique through a look-ahead scheme that smoothed the feedrate between the joint of two curve segments.
3. Implement an integrated look-ahead dynamics-based algorithm by considering contour and servo errors simultaneously.
4. Introduce a look-ahead trajectory generation to determine the acceleration stage according to the fast estimated arc length and the reverse interpolation of each curve segment.
5. Create a real-time look-ahead scheme that combines of path-smoothing, bi-directional scanning and feedrate scheduling.

[?] - Development of Real-time Look-Ahead Algorithm for NURBS Interpolator with Consideration of Servo Dynamics

[?] Development of a dynamics-based NURBS interpolator with real-time look-ahead algorithm

[?] - Development of a NURBS Curve Interpolator with Look-ahead Control and Feedrate Filtering for CNC System

[?] - This paper presents a perfect tracking method by combining iterative learning control (ILC) with disturbance observer (DOB) for CNC machine tools that perform the same tasks repeatedly. Although CNC systems have many nonlinear factors, they can be easily solved by ILC instead of complicated modeling. Also, for repeated disturbance, ILC performs very well and for non-repeated disturbance, DOB works much better. Besides, in order to facilitate the application, this paper proposes a variable gain algorithm with conditional compensation. Simulative results demonstrate the proposed learning scheme can improve machining accuracy when the CNC machine tools perform repetitive machining tasks.

## 2.9 Feedrate Filtering

The function of feedrate filtering is to obtain a continuous acceleration profile and avoid sudden changes in acceleration/deceleration during machining. The function of the adaptive feedrate adjustment is to adjust the feedrate, confine chord error and make acceleration meet the capabilities of the machine.

To obtain a continuous feedrate and acceleration during the whole interpolation process, one method is feedrate filtering [13]. This method adaptively adjusts the feedrate according to the different curvatures to meet the demand of the CNC machining accuracy.

Another popular method to eliminate jerky motion is spline interpolation [13-18]. In spline interpolation, the tool moves smoothly from one point to another. For example,



in Non-Uniform Rational B-Spline (NURBS) interpolation, the curve is parametrized and approximated by a set of interpolated data points, consisting of control points, knot points and weights.

NURBS parametric interpolation is often preferred over conventional linear and circular interpolator for its merits in terms of the model representation, feedrate smoothness and application range. The parametric curve is smooth and continuous so feedrate continuity is achieved effectively since the sharp junctions between curve segments are avoided.

Other methods include constant feedrate interpolation algorithms based on first-order, second-order Taylor expansions, and adaptive interpolation algorithm with confined chord error [10,12,22]. A feedrate fluctuation compensating algorithm that predicts and compensates feedrate fluctuation in real-time, has also been considered [18].

[?] Develop on feed-forward real time compensation control system for movement error in CNC machining  
 [?] Feedrate fluctuation compensating NURBS interpolator for CNC machining  
 [?] - Development of a NURBS Curve Interpolator with Look-ahead Control and Feedrate Filtering for CNC System

## 2.10 Realtime and Parallel Computing

### Parallel

[?] A parallel CNC system architecture based on Symmetric Multi-processor

### Realtime

[?] Develop on feed-forward real time compensation control system for movement error in CNC machining

[?] A real-time scheme of cubic parametric curve interpolations for CNC systems

[?] - Development of Real-time Look-Ahead Algorithm for NURBS Interpolator with Consideration of Servo Dynamics

[?] C/C++ and Python for Linux Realtime Parallel Port Software Driver

[?] Development of a dynamics-based NURBS interpolator with real-time look-ahead algorithm

[?] Real-time curve interpolators

[?] A real-time scheme of cubic parametric curve interpolations for CNC systems

[?] Reduction of the Contouring Error in High-Feed-Speed Machining by Real-Time Tracking-Error Compensation

Various researchers have implemented realtime CNC software systems on Windows platform running its RTX realtime operating system [26]. Free and open source CNC software are also available for the Linux platform through RT-Linux [27] realtime operating system (RTOS) or Real-Time Application Interface (RTAI) extensions.

The LinuxCNC [31] system is an open source RTAI-based CNC machine controller. It can drive milling machines, lathes, 3d printers, laser cutters, plasma cutters, robot arms, hexapods, and more. The controller for LinuxCNC is named EMC2 (Enhanced Machine

Controller version 2).

To take advantage of multi-cores and multi-processors available on personal computers, several implementations for high-speed CNC interpolation scheme using parallel computing have also been conducted.

In one method, the CNC interpolation method was divided into two tasks, the rough task executing in the personal computer and the fine task in the I/O card. During the interpolation procedure, double data buffers were constructed to exchange interpolation data between the two tasks in parallel [28].

In another method, parallelization of the CNC algorithm was implemented using multi-threading based on the Pthread software programming library [29]. Another variation is the method where a parametric curve realtime CNC interpolator processing was executed in parallel using the C++ Open Multi-Processing (OpenMP) Application Program Interface (API) library [30].

## 2.11 AI Approaches in CNC Machining

[?] Embedded Iterative Learning Contouring Controller Based on Precise Estimation of Contour Error for CNC Machine Tools

[?] The design of iterative learning control scheme for CNC machine tools

[?] CNC Interpolators: Algorithms and Analysis

[?] Interpolator for a CNC System

[?] The Error Analysis of Surface Machining on the CNC Machine Tools

[?] Adaptive Error Control Algorithm for High Speed Two-Axis CNC Contouring

## 2.12 Commercial versus Open CNC systems

[?] Frame Work of LV-UTHM: AN ISO 14649 Based Open Control System for CNC Milling Machine

## 2.13 Embedded devices in CNC systems

[?] Using Raspberry Pi 3 Model B to drive a CNC System in Real Time

[?] C/C++ and Python for Linux Realtime Parallel Port Software Driver

[?] Using Arduino Due to drive a CNC System

[?] Using the Nexys-3 Spartan-6 FPGA board to develop a closed-loop feedback CNC system

[?] A CNC machining system for education

[?] On the approach to CNC machining simulation improving

[?] Embedded Iterative Learning Contouring Controller Based on Precise Estimation of Contour Error for CNC Machine Tools



## 2.14 Summary on Literature Survey

# 3 Research Methodology

## 3.1 Research Objectives

The proposed research is to implement a realtime and parallel look-ahead control and feedrate compensation strategy for CNC reference-pulse interpolation.

Interpolation is the task that generates the actual reference commands that drives the CNC tool along the different axis-of-motions in the machine, such that, it accurately follows the desired machining path, in a timely and coordinated manner.

### 3.1.1 Research scope

The scope of work proposed for this research study are as follows:

1. Start from the provision of G-Codes, specifically RS274D NGC standard G-Code.
2. Implement reference-pulse CNC interpolation for computational efficiency.
3. Conduct look-ahead and feedrate error compensation in G-Code interpolation.
4. Execute realtime, parallel, online/offline computations for the CNC control loop.
5. Compare appropriate parallel execution methods for 2D/3D G-Code interpolation
6. Address designs for extension to 3-axis and 5-axis interpolation implementations.

### 3.1.2 G-Codes Coverage

We will start from RS274D NGC G-Code files because the format is the base standard for G-Codes supported by all machines used in the CNC industry. Our research does not include SVG or STL files generated by CAD applications. Our research 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, NURBS G-Code will not be in the scope of this research. It should be noted that NURBS G-Code files are not the same as NURBS interpolation method.

### 3.1.3 Reference-Pulse Interpolation

We will implement CNC Reference-Pulse Interpolation instead of Reference-Word interpolation because of computational efficiency. All reference-pulse interpolations are iterative and controlled by an adjustable interrupt clock. Various free and open source software libraries can take advantage of this fact and that makes integration practical. This integration method is also more streamlined with the reference-pulse method. Various implementation methods using these libraries can be explored for computational efficiency.

### 3.1.4 Look-ahead control

The feedrate look-ahead control algorithm is introduced to deal with sudden changes of feedrate pulses, essentially the move velocity. Look-ahead control is a pre-processing task of the path contour before real machining takes place. The main goal is to achieve a smooth feedrate (velocity) profile by looking ahead at deceleration regions in the given contour path. The idea of look ahead is to consider velocities to move toward future target points from the current machine position using the geometric path data already provided in the G-Code. That could be one-step forward, or two-step forward look-ahead methods. Using look-ahead control, the movement velocity can then be adjusted continuously to ensure smooth contouring movements, not jerky.

### 3.1.5 Feedrate compensation

Feedrate compensation is also about adjusting move velocities but with a focus on reducing contouring path errors, instead of maintaining movement smoothness. The two goals are conflicting, meaning, to reduce path errors we need to slow down the machine move velocity. Whereas, to speed up machine movements, we may increase path tracking errors. In this research, we will look at methods that strike a balance in minimizing contour errors, meaning, accurately following the desired machining path, and at the same time achieving a reasonable machining velocity. The term feedrate compensation or velocity compensation is used here to refer to velocity adjustments that have to be made. The target priority is placed on contour tracking accuracy compared to fast machining completion and velocity profile smoothness. Ultimately, there is no point to quickly finish cutting a part when the output machined part itself is not accurately machined.

### 3.1.6 Realtime and parallel execution

As discussed in Chapter 1, Introduction, our technical concept of realtime execution means every software task must be completed within its start and end timing deadlines, unlike the layman notion of realtime which means running in the current instance of time. In terms of parallel tasks execution, we will consider both software multi-threading and multi-processing methods. Our research strategy is to implement, wherever possible, both task-realtime and task-parallel executions simultaneously.

### 3.1.7 Parallel execution strategies

There are various parallel execution strategies that will be considered in CNC interpolation. The signals-and-slots (sigslot) mechanism is one execution strategy that will be used to allow different CNC software modules to inter-operate by calling each other's built-in functions. The sigslot method implements a type-safe, thread-safe signal/slot mechanism in software execution. The parallel executions for the signals-and-slots (signal/slot) method will be discussed further in the section on research implementation plan.

For example, in the CNC control software (CCS), the different software controller classes must be aware about each other in some detail. The centerpiece or crux of our methodology in the implementation of the CNC Control software is to design and execute the signal/slot algorithm that handles the different software classes through the CNC control loop.

The invocation of these classes must happen in a coordinated and timely manner, such that many tasks will be seen running in either serial, parallel or concurrent modes. This inter-operation, signal/slot, and control-loop strategy is the key to the success of our research on CNC interpolation. As discussed in Chapter 1, Introduction, the controller classes handled by the control loop comprise the following:

1. G-Code Interpreter class
2. G-Code Interpolator class
3. Signal Driver class
4. Motion Controller class
5. Error Controller class
6. Services Controller class
7. Human-Interface Controller class

### 3.1.8 Extension to 5-axis interpolation

The proposed research covers the 3-axis CNC Research machine, that we have available for development and experimentation on CNC control implementations. The CNC Research Machine and its details will be described in the next chapter, Chapter 4 on Related Research Work, Section 4.1 with a link here [??]. We will design the software control infrastructure to be ready for future inclusion and extension that cover 5-axis CNC machines.

## 3.2 Research Methodology

### 3.2.1 Software engineering perspective

Most of the work in CNC interpolation in published literature are conducted by control and instrument engineers, in which, the perspective is on the control and operational aspects of the CNC machine. We undertake this research project from the perspective of a software engineer, with a focus on exploring different software engineering technologies that can be applied to efficient CNC control operations.

### 3.2.2 Software engineering methods

Most of the work in CNC interpolation in published literature do not mention specific software programming techniques, data structures, and various tools, that are currently available for use by the software engineer. For CNC interpolation generally, the focus is direct to the mathematical aspects on geometry, path contours, and machine dynamics. Despite having common mention of realtime in CNC literature, the meaning is not in the true sense the correct interpretation of software defined realtime. This subject was discussed earlier in Chapter 1, on Introduction. For CNC machines in particular, with the readily available multi-cores, multi-processors, and networked computers today, there are less that a handful of CNC specific publications that even mention the word parallel for CNC interpolation. Thus, the two areas of realtime and parallel, are opportunities we will explore in this research.

For realtime software executions, we will implement task time recordings for true realtime using the RTAI (RealTime Application Interface) C/C++ library. For true parallel software executions, we will implement tasks for multi-threading, multi-processing, and parallel file HDF5 I/O operations using the OpenMPI (Open Message Passing Interface), OpenMP (Open Multi-Processing) and HDF5 (Hierarchical Data Format) C/C++ libraries. For offline CNC interpreter and CNC interpolator executions, we will implement the HDF5 high speed file I/O system. For CNC control loop operations, we will implement the Signal/Slot mechanism that allows programmed invocations and inter-operations of the various controller classes handled by the control loop.

### 3.2.3 Linux, open source and free software

Most of the work in CNC interpolation in published literature do not use Linux, open source and free software. As expected, Microsoft operating system (OS) seemed to be popular among control and instrument engineers. Except for embedded systems which is micro-controller (MCU) based, most motion-control hardware boards with micro-processors (MCU or CPU) for CNC use are Microsoft OS based. As we discussed in Chapter 1, Introduction, we avoided the Microsoft (MS) Windows platform because we need full and open access to the operating system and to the software components that control the devices attached to the operating system. With the Linux OS, we have full and open access to all software codes.

Our implementation for the CNC control loop and all its component controller classes shall be based primarily on the C/C++ software programming language. We will consider also the Rust programming language as the low-level systems language on par with C/C++. For scripting languages, we will consider Python and Julia to execute non-critical codes in the control loop. For complex mathematical computations, we will consider Octave-NURBS and Scilab-NURBS backend components that will be interfaced to the C/C++ primary control loop codes.

The software components, Rust, Python, Julia, Octave and Scilab are all open source and cost-free. In addition, all of the mentioned components can run in true parallel mode. For this research project, we have practically opted for total open access to all software components and total cost-free software.

### 3.2.4 Motion control devices

We will implement the Pico Universal PWM (Pulse Width Modulation) Servo Controller hardware board as our motion controller interface board to the 3-axis servo-controlled CNC Research machine. This is a LinuxCNC based interface board, that can handle the control of a 2-axis, 3-axis or 4-axis machine tool with PWM-driven servo amplifiers. It contains 4 PWM generators with variable PWM drive frequency (variable feedrate), and 4 digital encoder counters as feedback to follow the machine position. The image and specifications of the board can be seen at the link [??] and Fig. [??] in the appendix.

As comparison, we will implement the 28-Pin LIN (Local Interconnect Network) Demo Microchip Board for PIC16F/PIC18 MCUs as our own-developed interface board to drive the 3-axis servo-controlled CNC Research machine. The PIC Micro-controllers can be programmed from the Linux based MPLAB IDE software application provided by Microchip

Corporation. We have sufficient familiarity through previous successful experiences in programming PIC Microcontroller chips. The image and specifications of the board can be seen at the link [??] and Fig. [??] in the appendix.

In another variation, we will implement the Microchip Curiosity Development Board Demo for micro-controllers (MCU) as another own-developed interface board to drive the 3-axis servo-controlled CNC Research machine. Similarly, the MCUs can be programmed from the Linux based MPLAB IDE software application provided by Microchip Corporation. This Curiosity Demo Board allows for some user prototyping but not the full prototyping as provided in the 28-Pin-LIN Demo Board. The image and specifications of the board can be seen at the link [??] and Fig. [??] in the appendix.

Using our own developed CNC Control software, we will compare performance as well as limitations of all three board interface devices for driving the 3-axis servo-controlled CNC Research machine.

### 3.2.5 Research Validation

It is generally quoted that, *"The proof of the pudding is in the eating."* People say this to mean that something can only be judged to be good or bad after it has been tried or used.

For contour tracking error, we will compute the deviation from G-Code reference commands by the encoder feedback mechanism from our CNC Research machine servo-motor. This can be accomplished using our Pico Universal PWM Controller hardware board which comes equipped with 4-nos of digital encoder counters as feedback to follow the machine position (X, Y, Z).

For validation of our machining control technique, we will take manual micrometer measurements of our machine part based on 2D models of common objects, like a circular disk, a semi-circular disk, a square plate, a rectangular plate, a pentagon plate or a hexagon plate, a star and so on. These measurements will be compared with the G-Code exact coordinates drawn for those simple models. We will test our CNC Control Software on a real CNC Laser Cutting machine available at UMP for precision cutting.

For realtime and parallel tasks execution, we will capture (dump) in nanoseconds, timing records for the tasks, and save the data in a file, like HDF5 binary files. Parallel I/O HDF5 data formats are used for fast serialization and parallelization of large datasets. The files will later be analyzed to validate actual realtime and/or parallel tasks performance against its design.

### 3.3 Summary on Research Methodology

We will start our research from the point of being provided with the RS274D NGC formatted G-codes. Next, we implement CNC Reference-Pulse interpolation based on the provided G-codes. This is followed by either the one-step forward, or two-step forward look-ahead methods to determine machine move velocity adjustments for smooth overall machining.

The next step is to perform feedrate or velocity compensation, an activity that strikes a balance between achieving contour tracking accuracy against machine movement smoothness. In implementing our CNC control loop execution, our strategy is to implement, wherever possible, both task-realtime and task-parallel executions simultaneously. In the CNC control loop, the parallel executions using the signals-and-slots (signal/slot) method will allow different CNC software modules to inter-operate by calling each other's built-in functions, in both synchronous and asynchronous modes.

It must be noted that the centerpiece or crux of our methodology in the implementation of the CNC control software is to correctly design and execute the signal/slot algorithm that handles the different software classes through the CNC control loop. We shall also design the software control infrastructure to be ready for future extension to cover 5-axis CNC machines.

We undertake this research project from the perspective of a software engineer, with a focus on exploring different software engineering technologies that can be applied to efficient CNC control operations.

Realtime and parallel executions are two opportunities we will explore in this research on CNC interpolation. In addition, for this research project, we opt for total open access to all software components and total cost-free software by including Linux based, open source and only cost-free software resources. For the delivery of the CNC control software product, we will consider inclusion of software components, like Rust, Python, Julia, Octave and Scilab, in addition to the base C/C++ codes.

In this research project, we will implement three(3) hardware interface devices to drive our CNC Research machine, comprising of the Universal PWM (Pulse Width Modulation) Servo Controller board from Pico systems, the 28-Pin LIN (Local Interconnect Network) Demo Board for micro-controllers (MCU) from Microchip Corporation, and the Curiosity Development Board Demo for micro-controllers (MCU) also from Microchip Corporation. Our goal is to develop our own CNC Control software, and use this software to compare performance as well as limitations of all three board interface devices in driving the 3-axis servo-controlled CNC Research machine.

For research validation on our developed CNC Control software, it will be judged as being good or bad after it has been tried or used. We will perform computational assessments, as well as manual measurements of real parts produced to compare them against G-Code exact coordinates and dimensions drawn for the source of some simple 2D models.

In overview, the proposed research is to implement a realtime and parallel look-ahead control and feedrate compensation strategy for CNC reference-pulse interpolation. The descriptions above summarizes the research methodology for this project.

# 4 Software Interpolator Design



# 5 Simulation Experiment

## 5.1 Selected Parametric Equations

The ten(10) 2D parametric curves covered in this work are:

1. Teardrop
2. Butterfly
3. Ellipse
4. Skewed-Astroid
5. Circle
6. AstEpi = Astroid + Epicycloid combination
7. Snailshell
8. SnaHyp = Snailshell + Hypotrochoid combination
9. Ribbon-10L
10. Ribbon-100l = 10 times scaleup of Ribbon-10L

The parametric equations describing each of the curves  $x(u)$ , and  $y(u)$  are provided in the next table. The independent parameter  $u$  is limited to

$$u \in [0.0, 1.0]$$

The curves were selected based on their different characteristics like closed loop curves, open ended curves, symmetric or non-symmetric about the x-axis and y-axis, and having concave or convex turns. The x and y dimensions (sizes) vary among the different curves.

The main objective of the selection criteria is to ensure that the interpolation algorithm for the parametric curve succeeds and does not break in all cases.

The results for the feedrates in machining the ten(10) curves show continuity, smoothness, with no abrupt jumps as the CNC machine traverse the entire curve from the start ( $u = 0.0$ ) until the end ( $u = 1.0$ ).

### 5.1.1 Teardrop parametric equation

#### No. 1 - Teardrop parametric curve

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

Closed loop

Overall Single loop

Reflection x-axis: non-symmetrical

Reflection y-axis: symmetrical

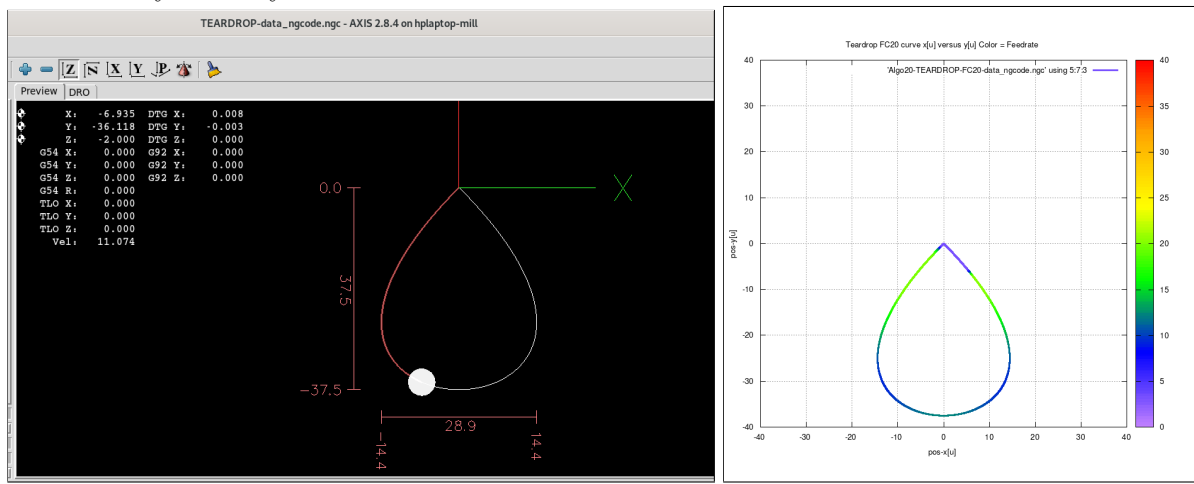


Table 5.1: Teardrop parametric equation and dimensions

### 5.1.2 Butterfly parametric equation

#### No. 2 - Butterfly parametric curve

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

Closed loop

Overall Multiple loops

Reflection x-axis: non-symmetrical

Reflection y-axis: symmetrical

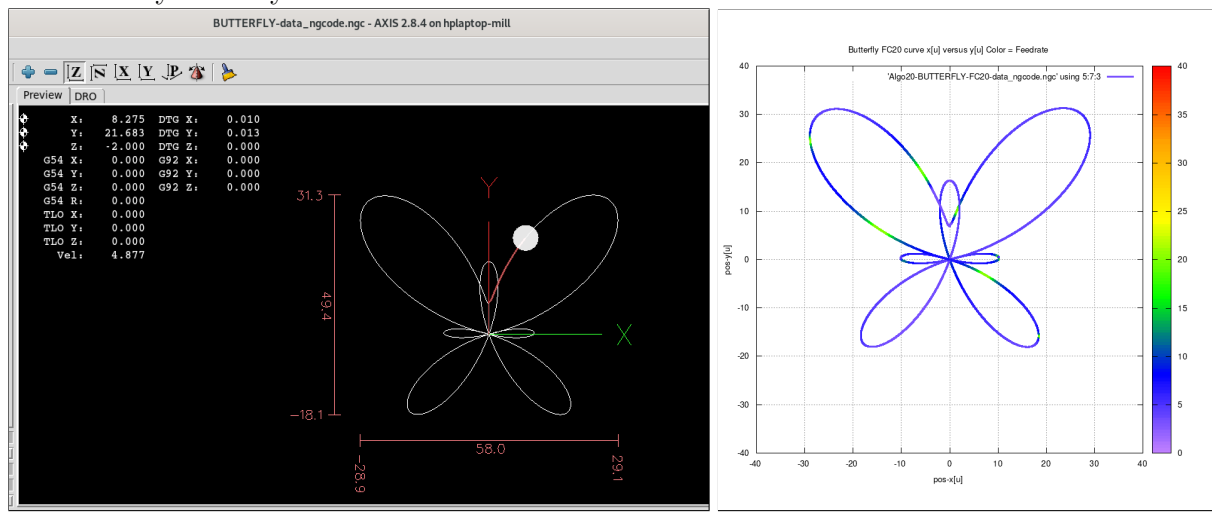


Table 5.2: Butterfly parametric equation and dimensions

### 5.1.3 Ellipse parametric equation

#### No. 3 - Ellipse parametric curve

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

Closed loop

Overall Single loop, smooth convex curves

Reflection x-axis: symmetrical

Reflection y-axis: symmetrical

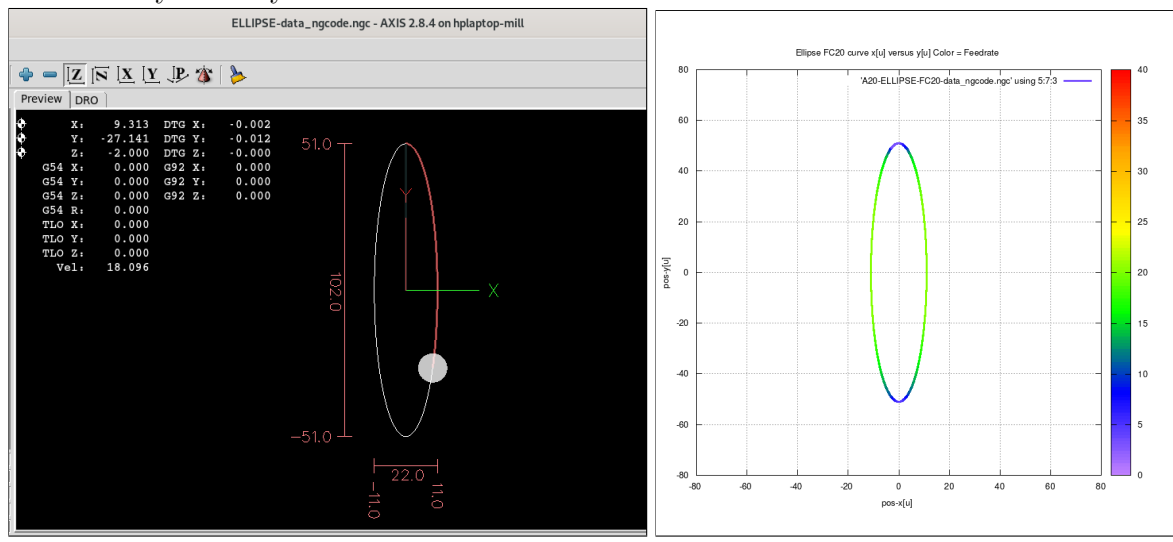


Table 5.3: Ellipse equation and dimensions

### 5.1.4 Skewed-Astroid parametric equation

#### No. 4 - Skewed-Astroid parametric curve

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

Closed loop

Overall Single loop, 4 cusps and 4 concave curves

Reflection x-axis: symmetrical

Reflection y-axis: symmetrical

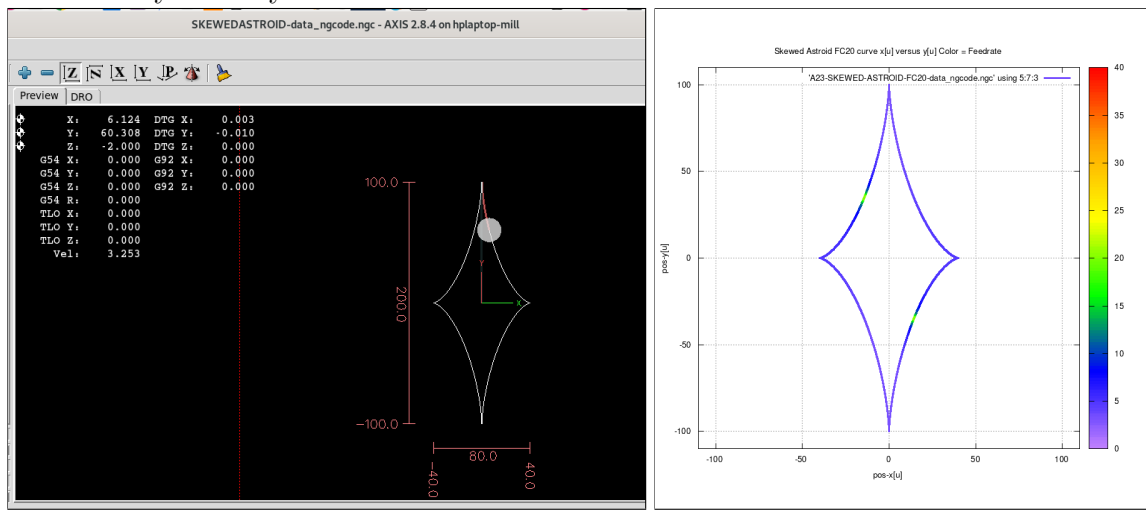


Table 5.4: Skewed-Astroid and dimensions

### 5.1.5 Circle parametric equation

#### No. 5 - Circle parametric curve

$$x(u) = 79 \sin(2\pi u)$$

$$y(u) = 79 \cos(2\pi u)$$

$$u \in [0.0, 1.0]$$

Closed loop

Overall Single loop, smooth convex curves

Reflection x-axis: symmetrical

Reflection y-axis: symmetrical

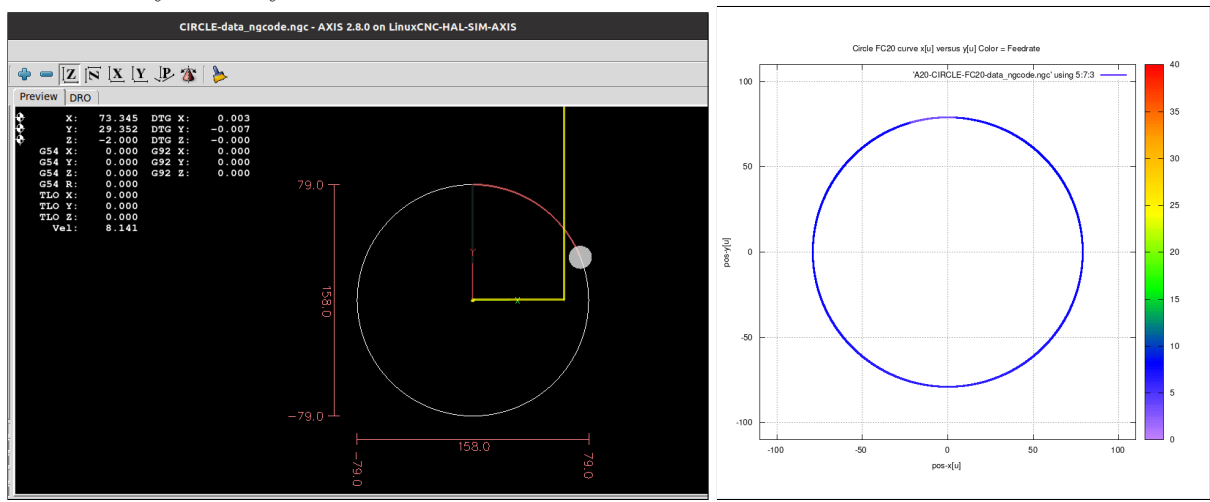


Table 5.5: Circle equation and dimensions

### 5.1.6 AstEpi parametric equation

#### No. 6 - AstEpi = Sum of (Astroid + Epicycloid) parametric curves

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

Closed loop

Overall Three loops, all convex curves

Reflection x-axis: non-symmetrical

Reflection y-axis: non-symmetrical

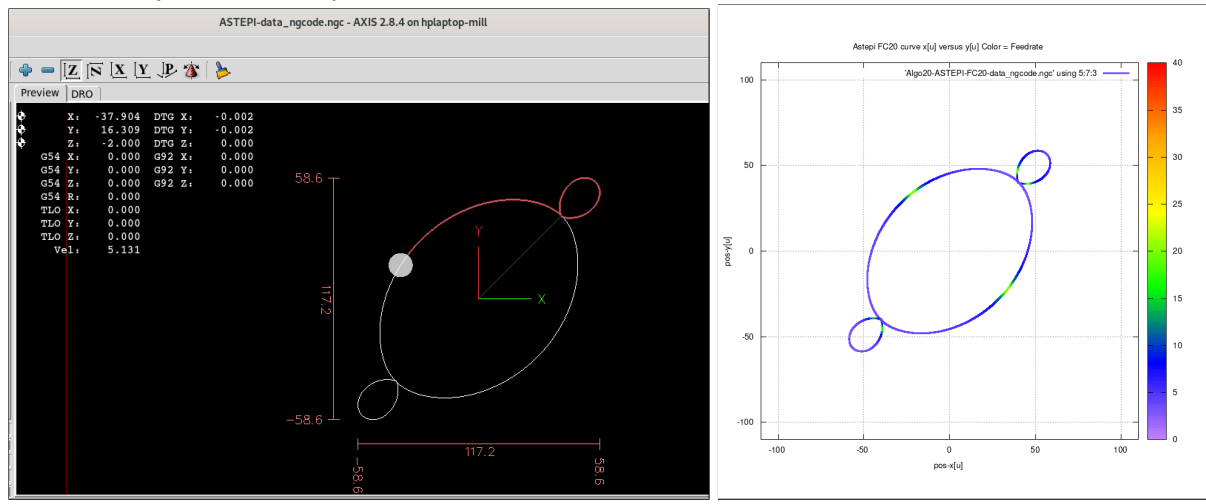


Table 5.6: Astepi equation and dimensions

### 5.1.7 Snailshell parametric equation

#### No. 7 - Snailshell parametric curve

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

Open ended curve

Overall No loop, smooth and continuous convex curves

Reflection x-axis: non-symmetrical

Reflection y-axis: non-symmetrical

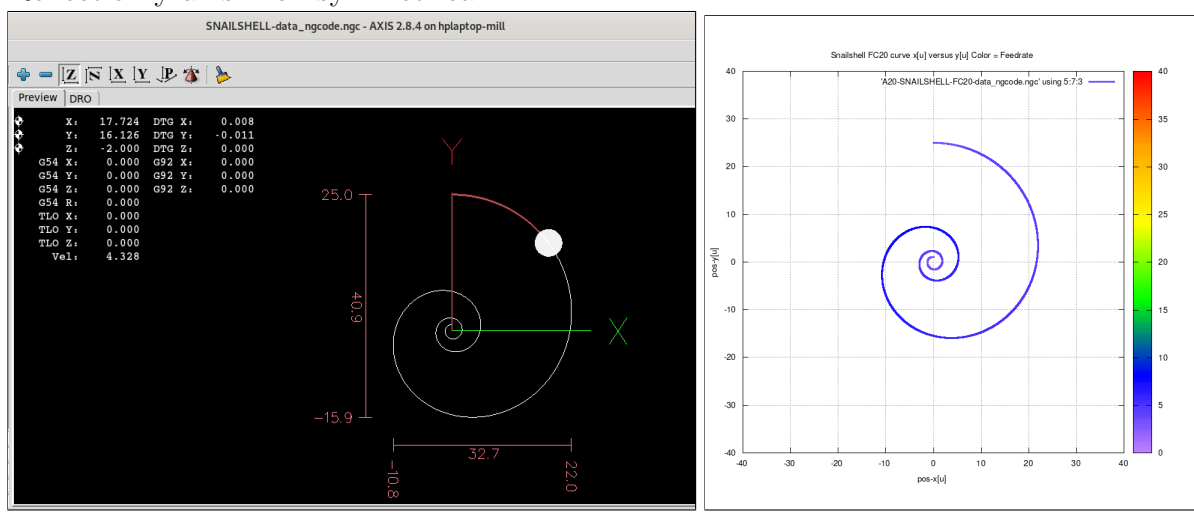


Table 5.7: Snailshell equation and dimensions



### 5.1.8 SnaHyp parametric equation

#### No. 8 - SnaHyp = Sum of (Snailshell + Hypotrochoid) parametric curves

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

Open ended curve

Overall 1 loop, except for 1 concave curve, the rest are convex curves

Reflection x-axis: non-symmetrical

Reflection y-axis: non-symmetrical

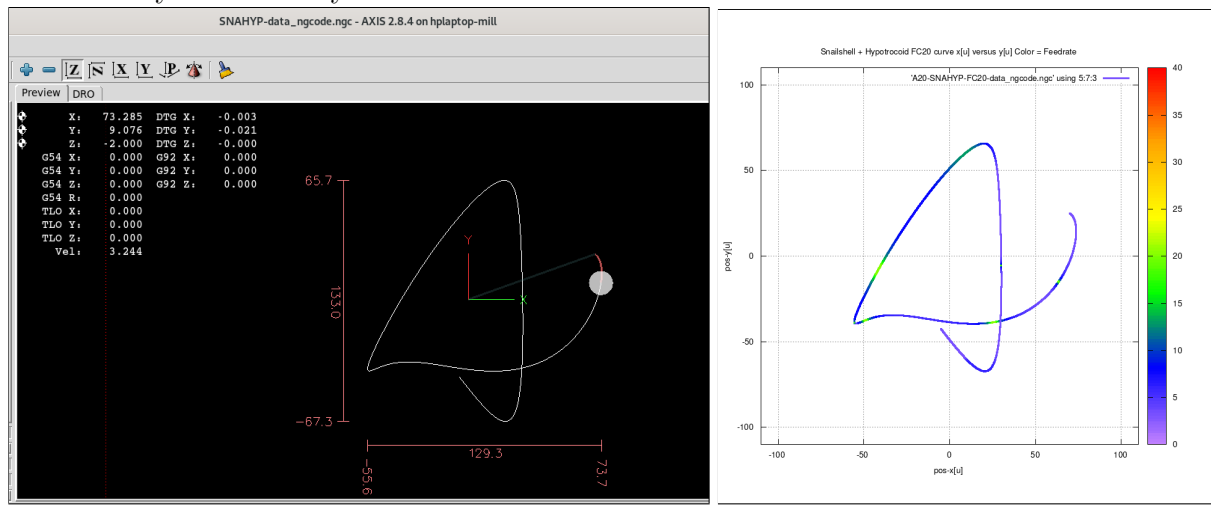


Table 5.8: SnaHyp equation and dimensions

### 5.1.9 Ribbon-10L parametric equation

#### No. 9 - Ribbon-10L parametric curve

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

Open ended curve

Overall Single loop, smooth convex curves

Reflection x-axis: non-symmetrical

Reflection y-axis: non-symmetrical

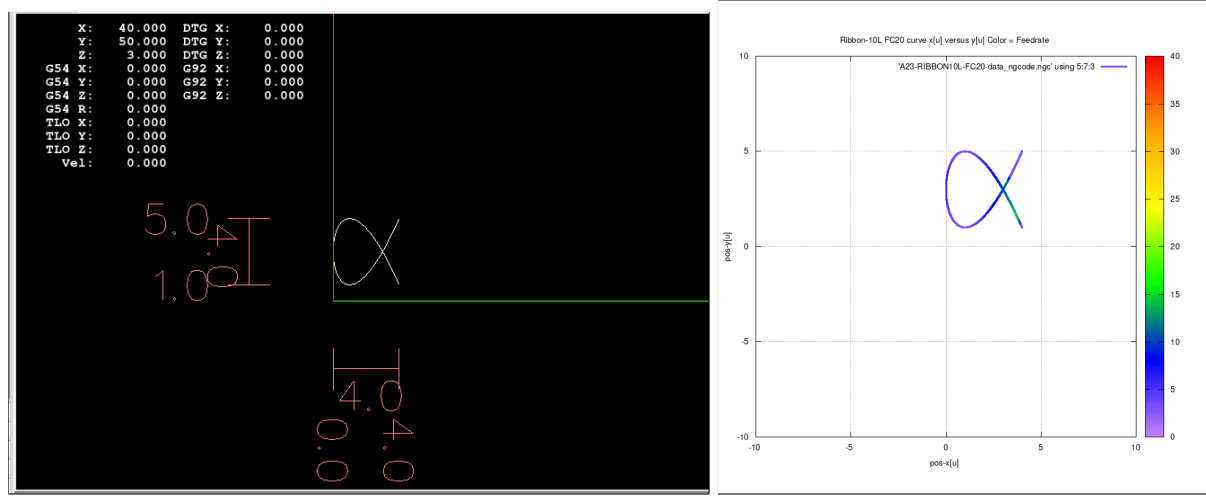


Table 5.9: Ribbon-10L equations and dimensions

### 5.1.10 Ribbon-100L parametric equation

#### No. 10 - Ribbon-100L parametric curve

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

Open ended curve (10 times larger than RIBBON-10L)

Overall Single loop, smooth convex curves

Reflection x-axis: non-symmetrical

Reflection y-axis: non-symmetrical

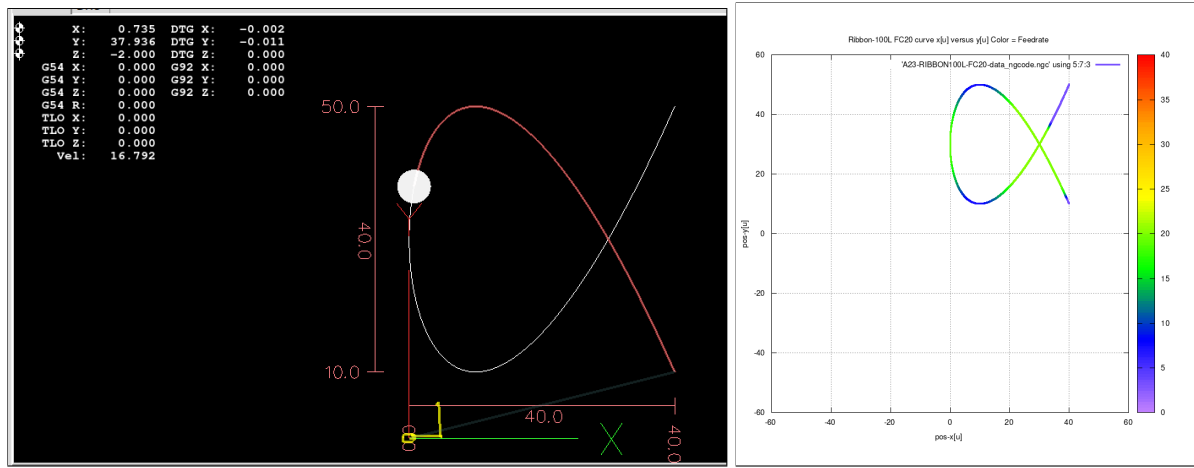


Table 5.10: Ribbon-100L equation and dimensions

## 5.2 Total Interpolated Points

### 5.2.1 Teardrop profile of interpolated points

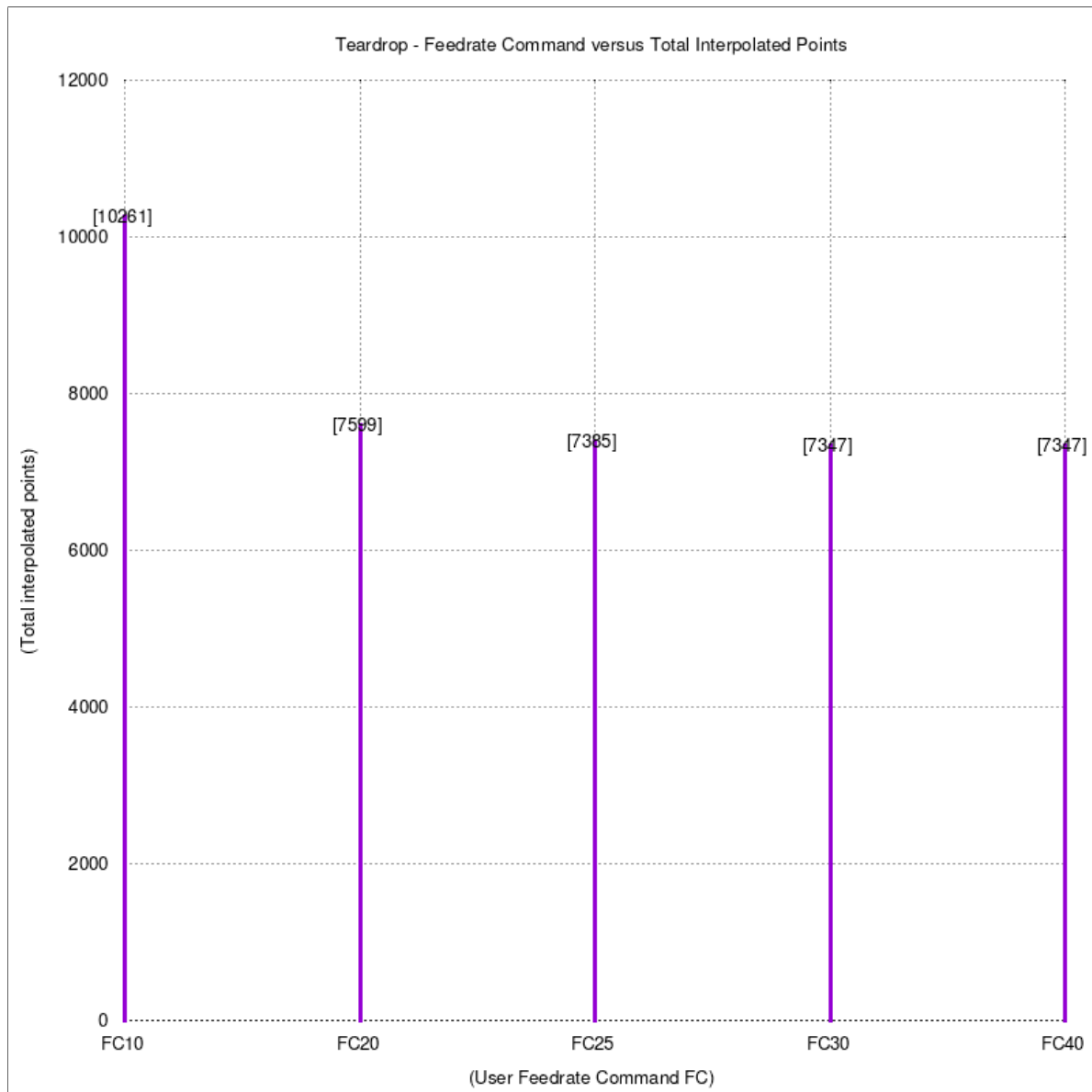


Table 5.11: Teardrop profile of interpolated points

### 5.2.2 Butterfly profile of interpolated points

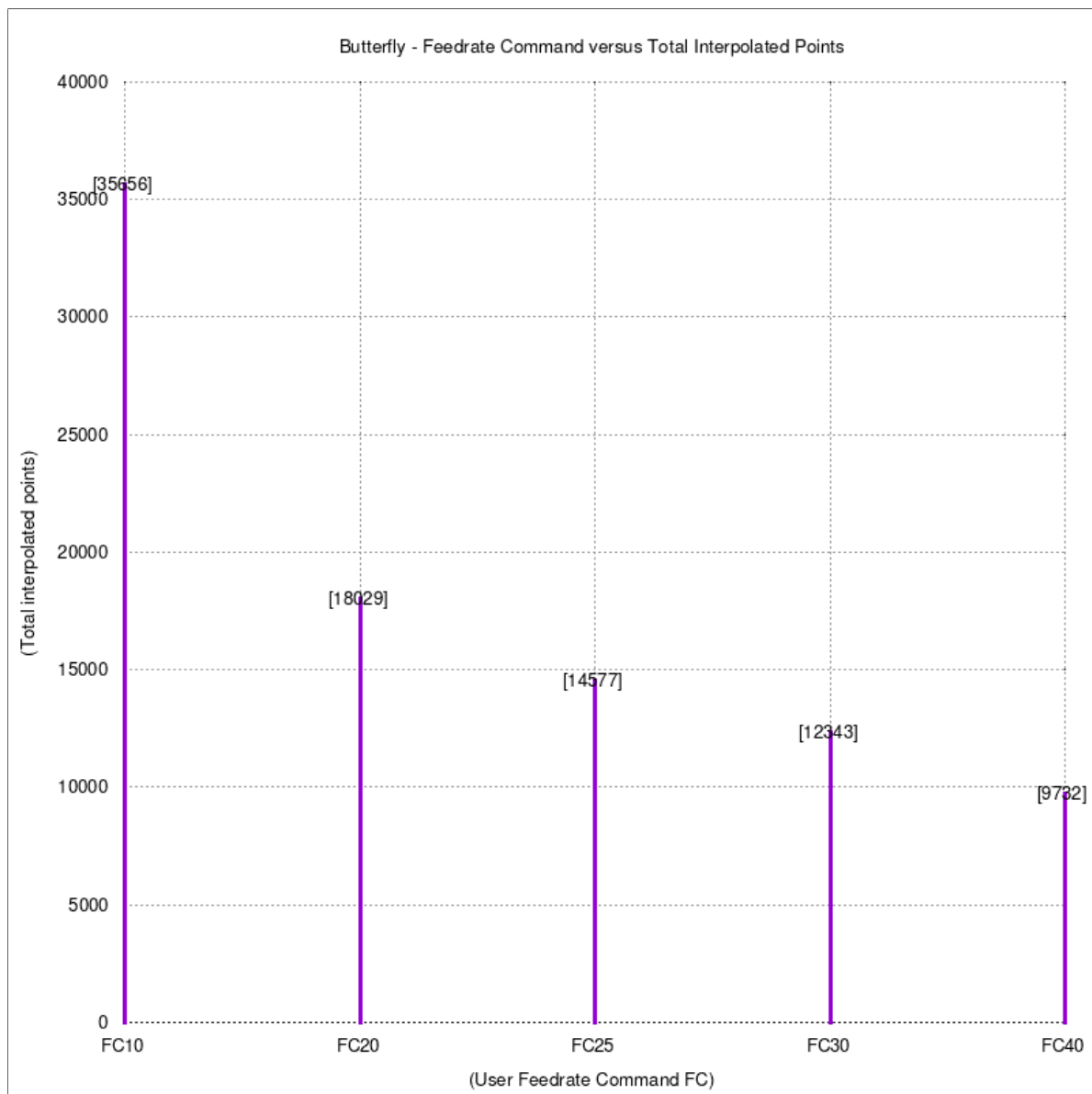


Table 5.12: Butterfly profile of interpolated points

### 5.2.3 Ellipse profile of interpolated points

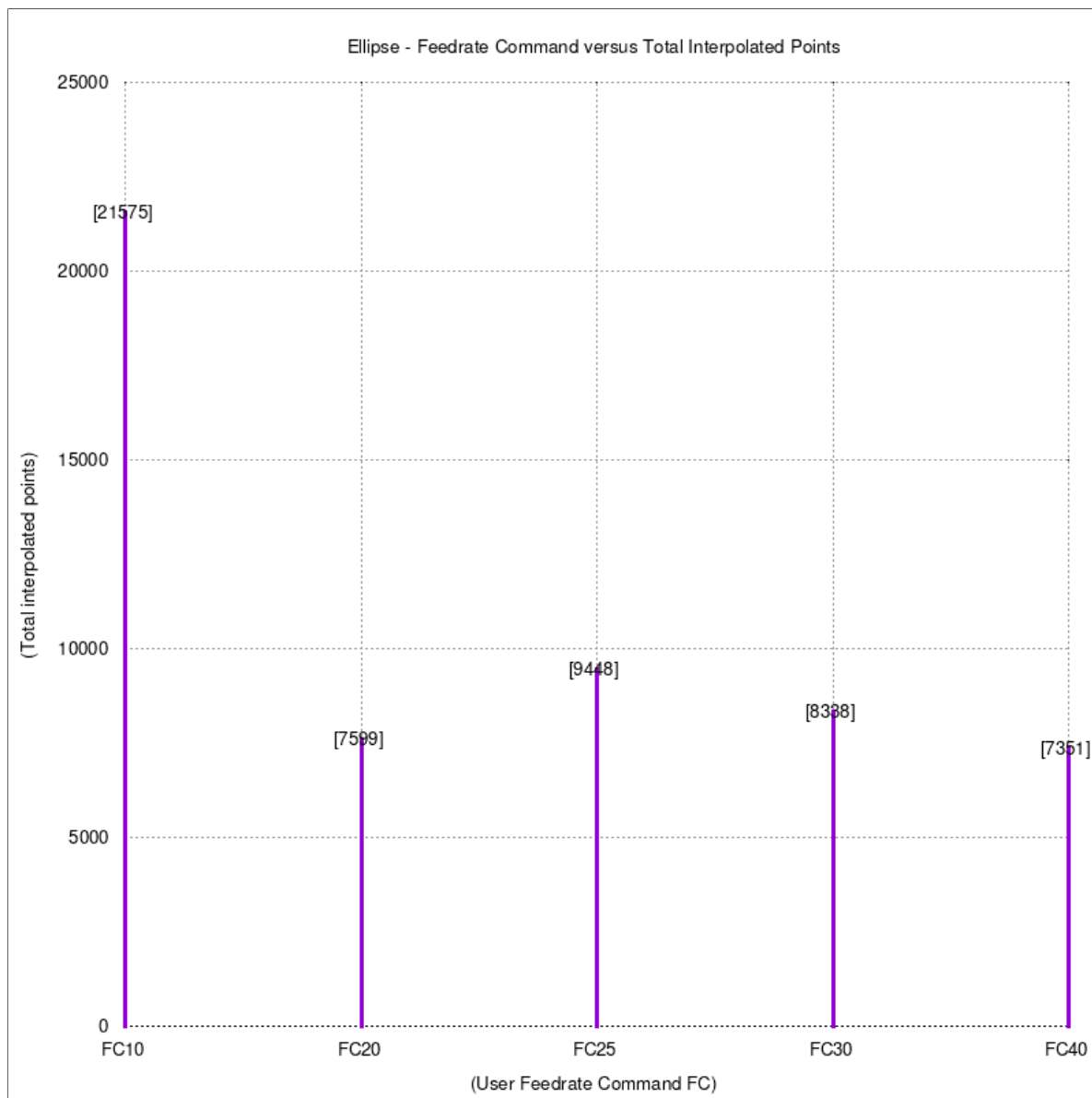


Table 5.13: Ellipse profile of interpolated points

### 5.2.4 Skewed-Astroid profile of interpolated points

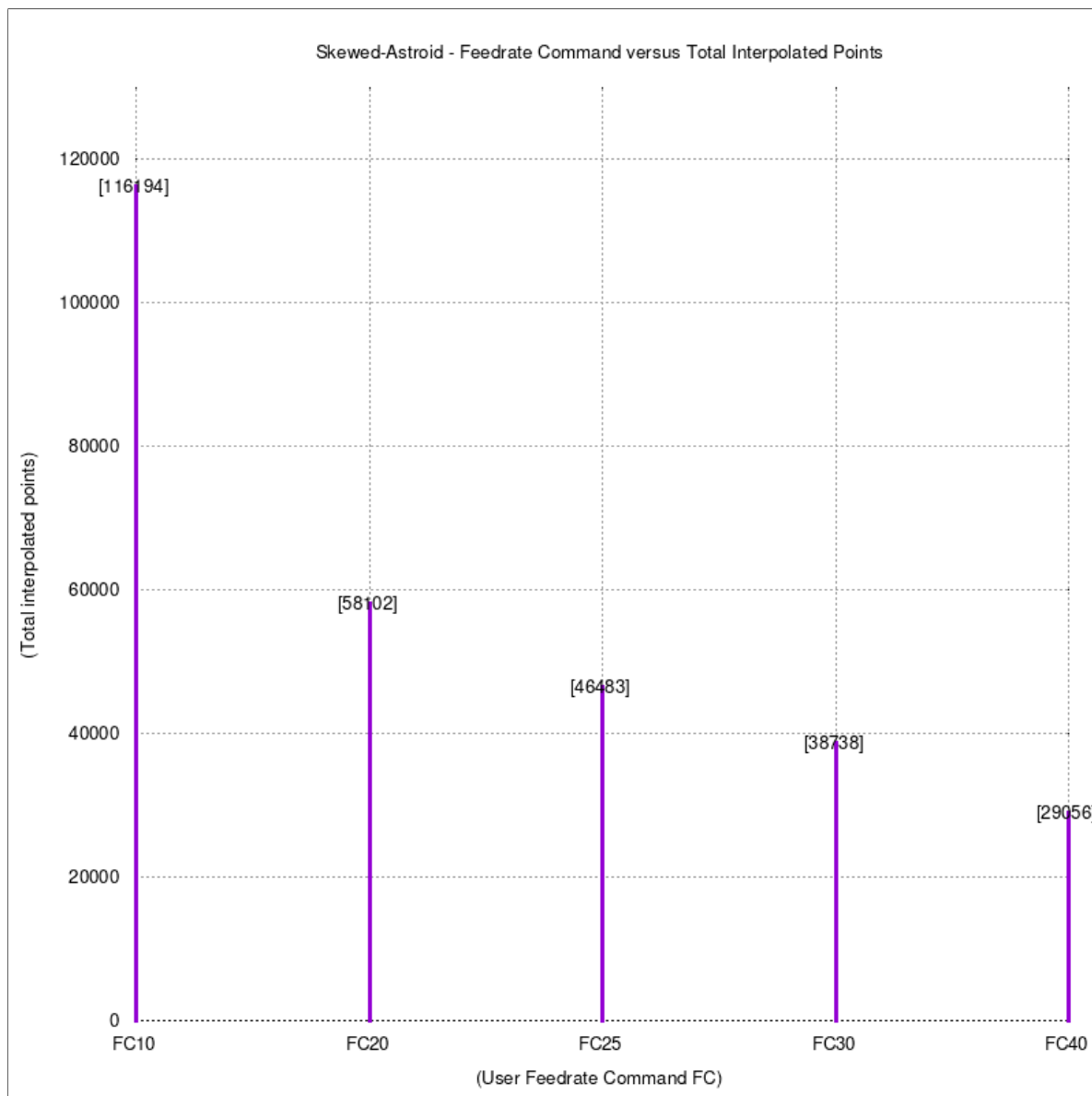


Table 5.14: Skewed-Astroid profile of interpolated points



### 5.2.5 Circle profile of interpolated points

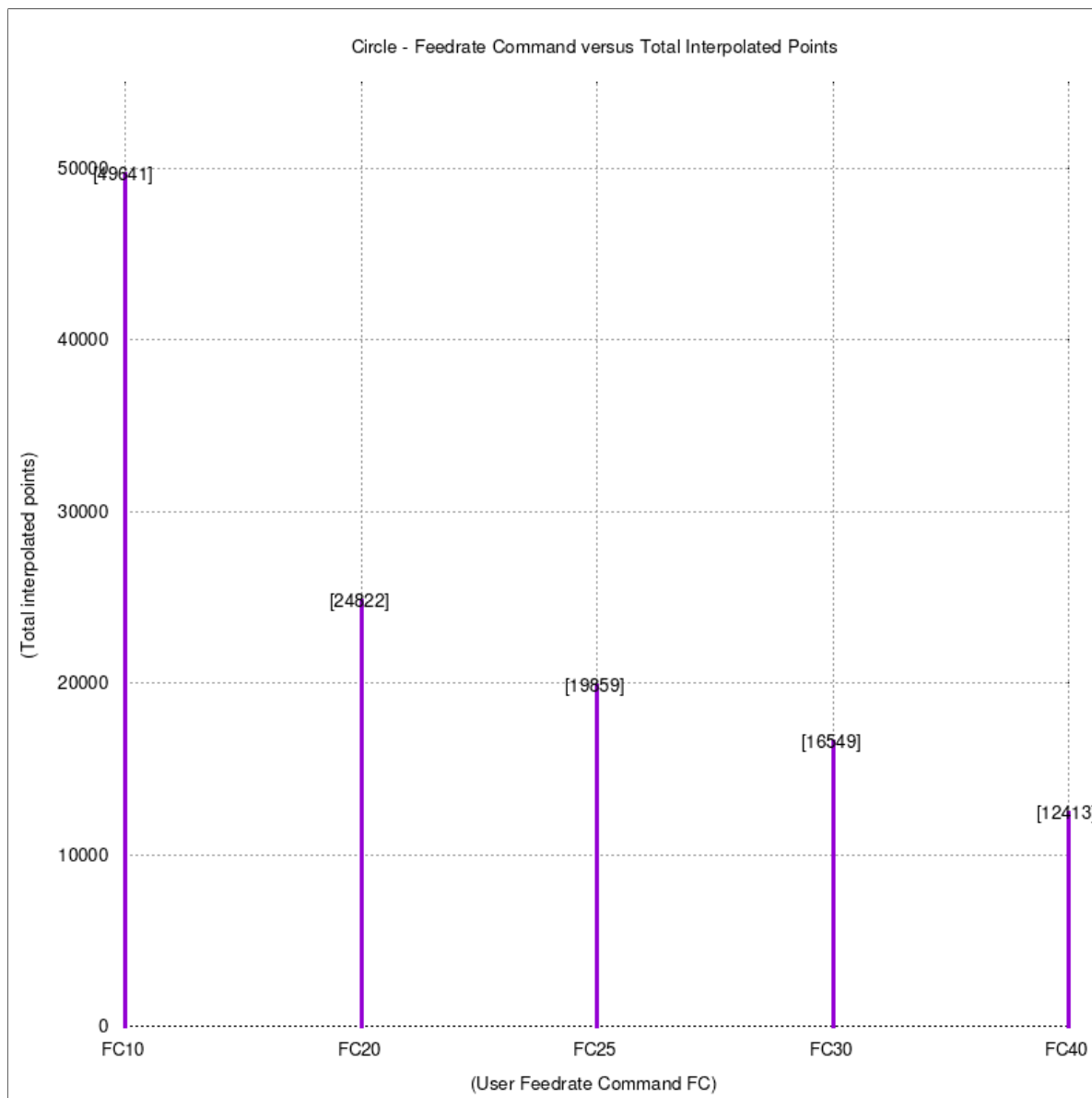


Table 5.15: Circle profile of interpolated points

### 5.2.6 Astroid-Epicycloid profile of interpolated points

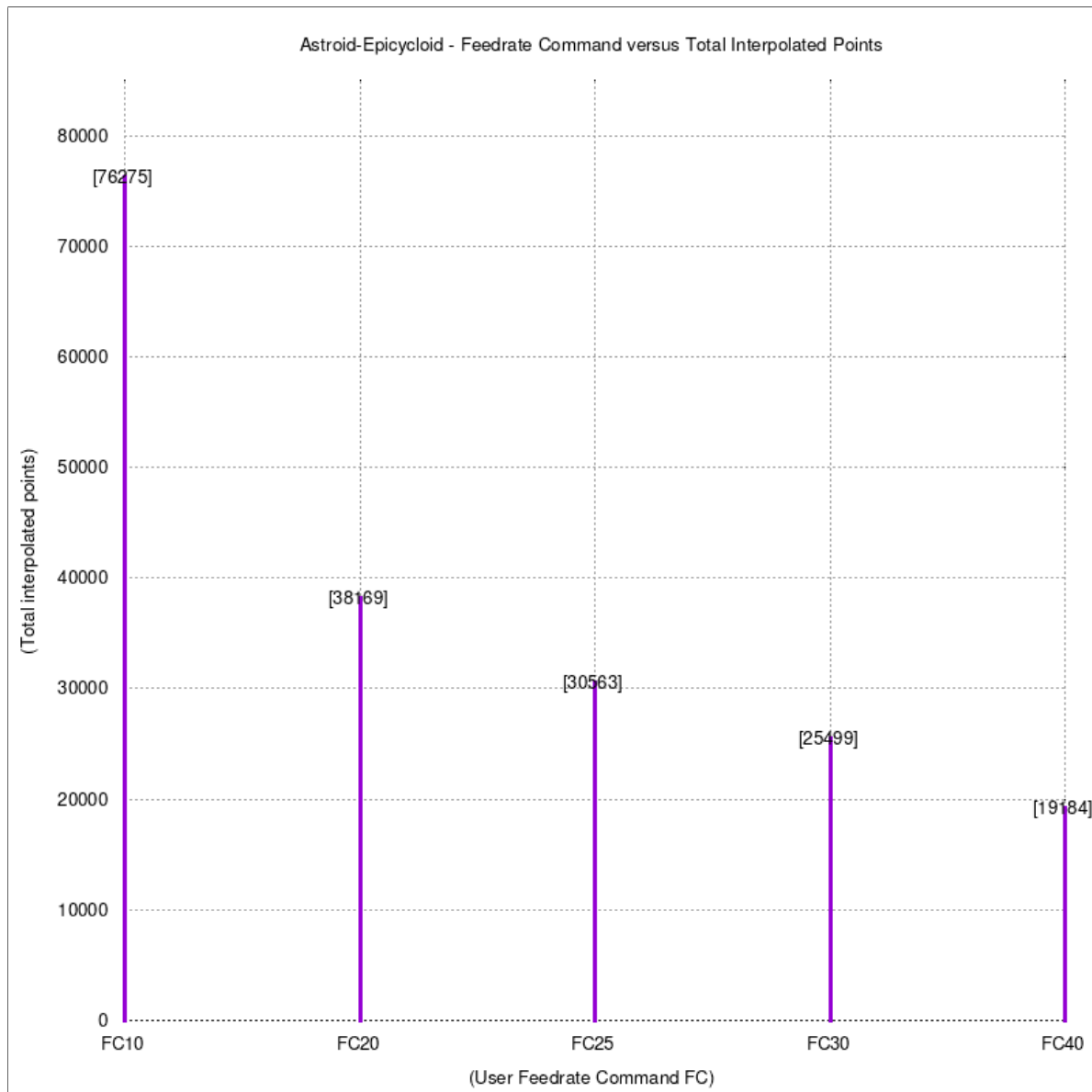


Table 5.16: AstEpi profile of interpolated points

### 5.2.7 Snailshell-Hypotrocid profile of interpolated points

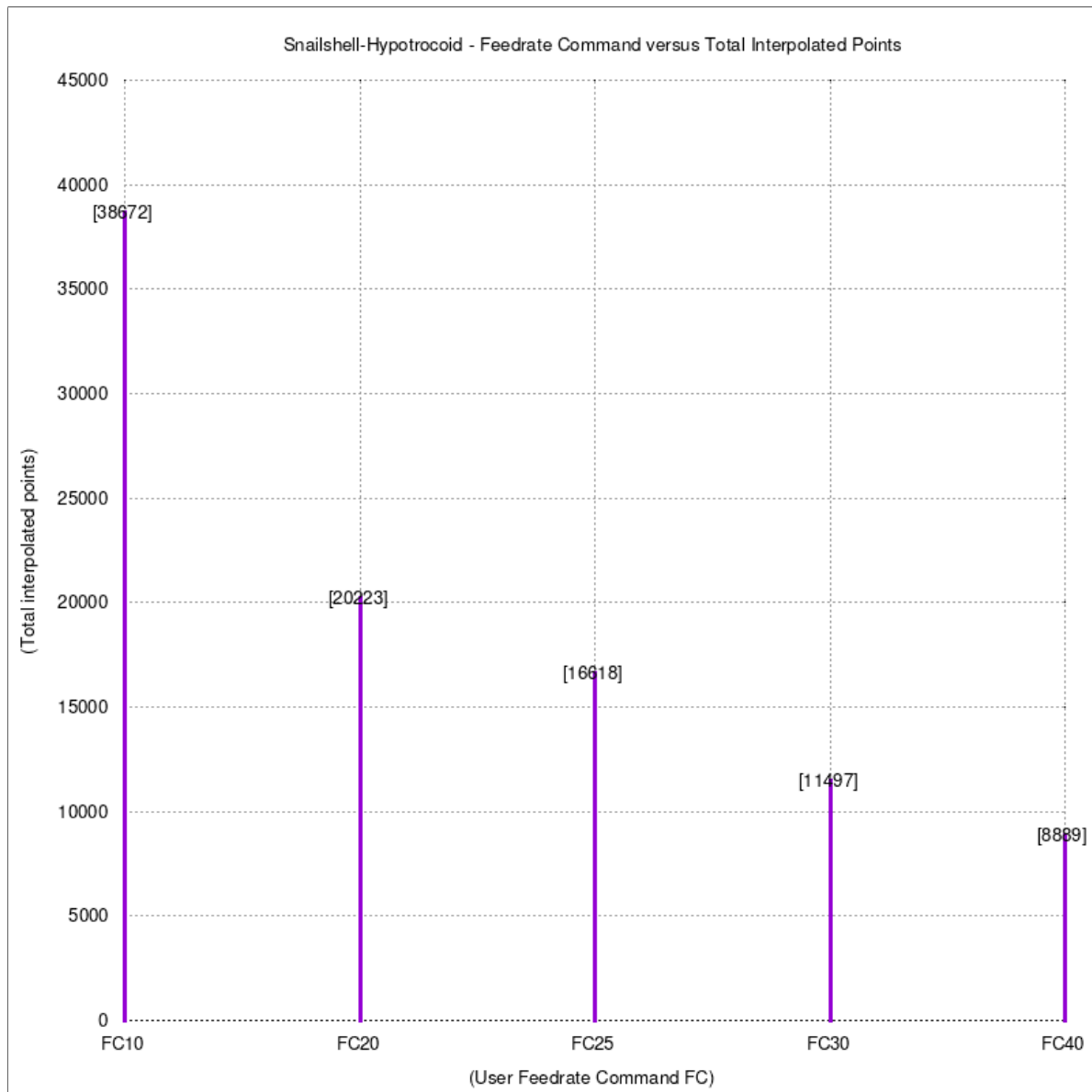


Table 5.17: SnaHyp profile of interpolated points

### 5.2.8 Snailshell profile of interpolated points

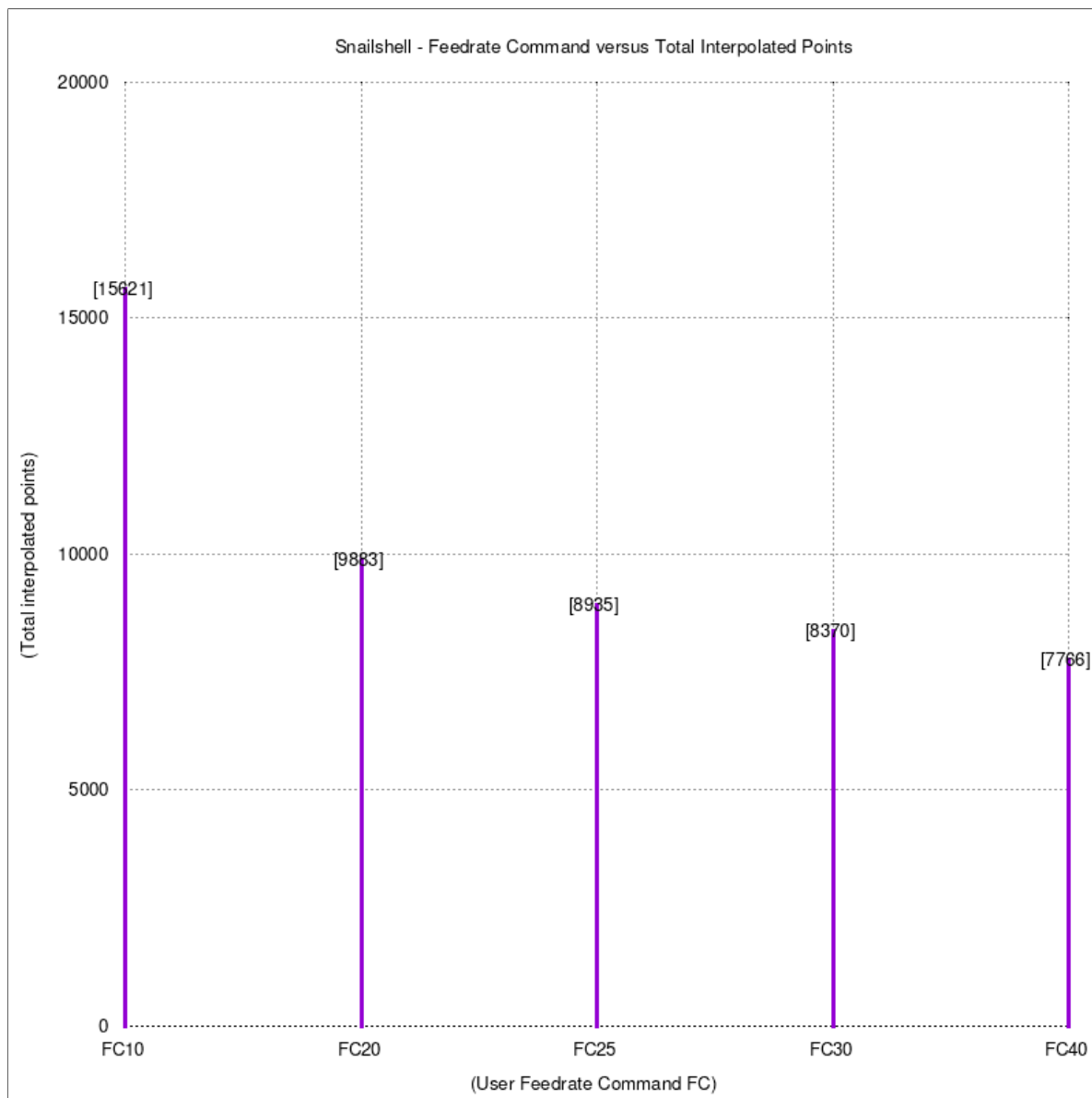


Table 5.18: Snailshell profile of interpolated points

### 5.2.9 Ribbon-10L profile of interpolated points

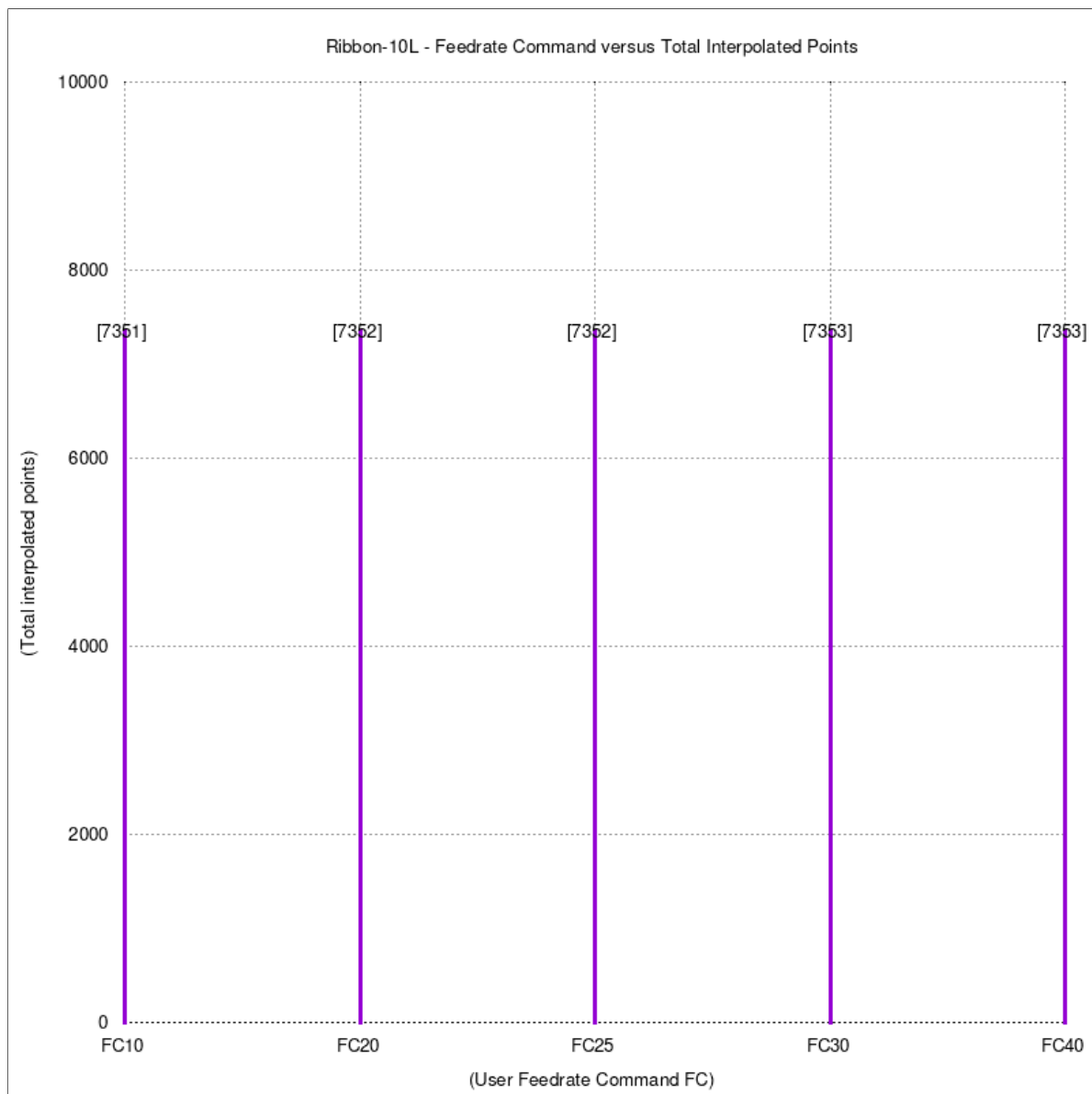


Table 5.19: Ribbon-10L profile of interpolated points

### 5.2.10 Ribbon-100L profile of interpolated points

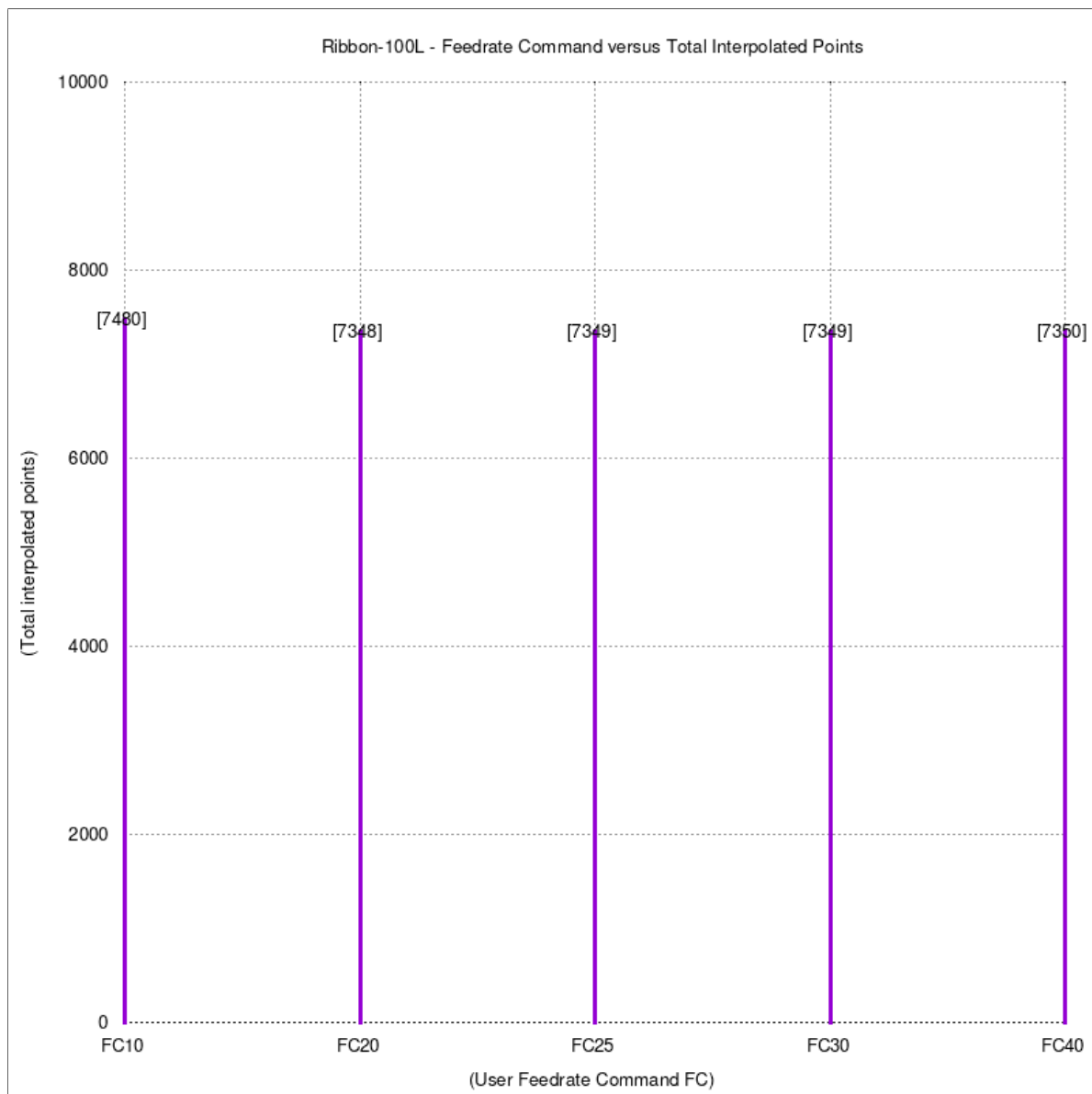


Table 5.20: Ribbon-100L profile of interpolated points

### 5.3 Experimental Run Results

Bismillah

Describe the Table FC10, FC, 20, FC25, FC30 and FC40

### 5.3.1 Teardrop and Butterfly Run Data

Date: 2023-06-06 Author: wruslandr@gmail.com		Report CNC Parametric Curve Interpolation and Trajectory Tracking Part 1 of 5 Teardrop and Butterfly (x-y) parametric curves									
ITEM	DESCRIPTION	TEARDROP CURVE					BUTTERFLY CURVE				
		FC10	FC20	FC25	FC30	FC40	FC10	FC20	FC25	FC30	FC40
1	Run user feedrate command (mm/s)										
2	Total interpolated u-points	10261	7599	7385	7347	7347	35656	18029	14577	12343	9732
3	Parameter completion (reached u-end)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
<b>Pushdown epsilon eps(u) algorithm</b>											
4	Count before pushdown, eps(u) is below (1E-6)	8498	1427	527	0	0	35465	17421	13602	11010	7256
5	Count pushdown points, eps(u) to below (1E-6)	1763	6172	6858	7347	7347	191	608	975	1333	2476
<b>Epsilon eps(u) chord error</b>											
6	Count eps(u) above (1E-6)	0	0	0	0	0	0	0	0	0	0
7	Count eps(u) in (1E-7, 1E-6)	10261	7599	7385	7347	7347	2995	12494	13794	12343	9732
8	Count eps(u) in (1E-8, 1E-7)	0	0	0	0	0	32661	5535	783	0	0
9	Count eps(u) in (1E-9, 1E-8)	0	0	0	0	0	0	0	0	0	0
10	Count eps(u) in (1E-10, 1E-9)	0	0	0	0	0	0	0	0	0	0
11	Count eps(u) below (1E-10)	0	0	0	0	0	0	0	0	0	0
<b>Count interpolated u-points</b>											
12	Count rising S curve u-points	960	480	389	370	370	1323	693	575	500	418
13	Count frate is lower than fratelimit	4734	4342	4260	4202	4049	17751	8968	7255	6129	4772
14	Count frate is equal to fratelimit	0	0	0	0	0	0	0	0	0	0
15	Count frate is higher than fratelimit	3608	2298	2348	2406	2559	15254	7673	6171	5213	4124
16	Count falling S curve u-points	959	479	388	369	369	1328	695	576	501	418
<b>Count u-points histogram (G01 codes)</b>											
17	Count u-points [0.00 <= u < 0.10]	1734	875	768	748	748	3463	1763	1431	1214	952
18	Count u-points [0.10 <= u < 0.20]	1120	791	791	791	791	4332	2167	1733	1444	1112
19	Count u-points [0.20 <= u < 0.30]	809	794	794	794	794	2983	1554	1287	1117	927
20	Count u-points [0.30 <= u < 0.40]	726	710	710	711	711	3220	1611	1293	1098	877
21	Count u-points [0.40 <= u < 0.50]	741	629	629	629	629	3832	1920	1545	1299	998
22	Count u-points [0.50 <= u < 0.60]	742	629	629	628	629	3829	1919	1544	1298	997
23	Count u-points [0.60 <= u < 0.70]	726	710	711	711	711	3222	1612	1294	1098	878
24	Count u-points [0.70 <= u < 0.80]	809	794	793	794	793	2981	1553	1286	1117	926
25	Count u-points [0.80 <= u < 0.90]	1120	791	791	791	792	4323	2162	1730	1441	1110
26	Count u-points [0.90 <= u <= 1.00]	1734	876	769	750	749	3471	1768	1434	1217	955
27	Check Total u-points	10261	7599	7385	7347	7347	35656	18029	14577	12343	9732
<b>Performance</b>											
28	Total curve error (sum of epsilon(u))	0.005809	0.007141	0.007301	0.007337	0.007335	0.001939	0.003534	0.004231	0.004847	0.005851
29	Total dist traversed (sum of chord lengths)	101.8357	101.8419	101.8348	101.8596	101.8356	356.0747	356.0737	356.0723	356.0728	356.0732
30	Percentage (Tot curve error / Tot dist traversed)	0.005704	0.007012	0.00717	0.007203	0.007203	0.000545	0.000993	0.001188	0.001361	0.001643

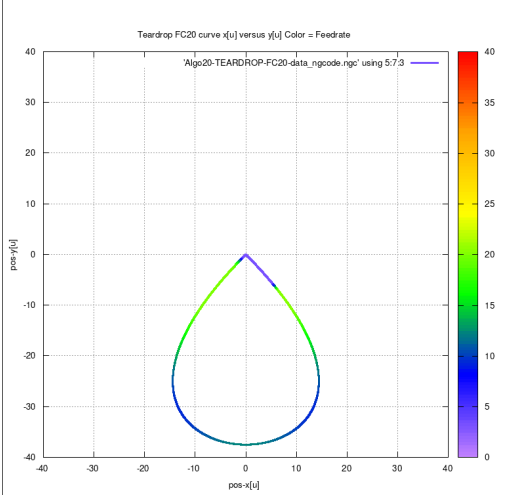
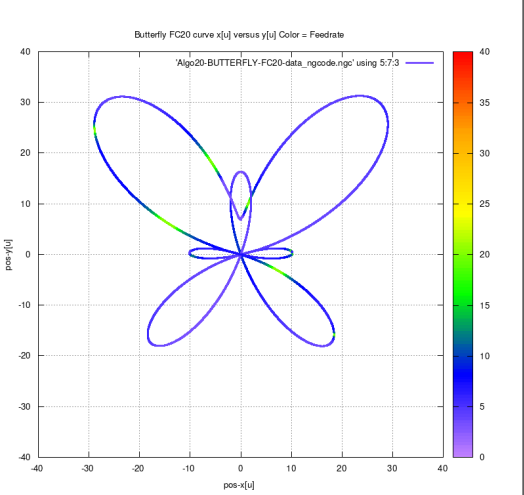



Table 5.21: Teardrop and Butterfly Run Data

Bismillah

Allah huakbar



### 5.3.2 Ellipse and Skewed-Astroid Run Data

Date: 2023-06-06 Author: <a href="mailto:wruslandr@gmail.com">wruslandr@gmail.com</a>		Report CNC Parametric Curve Interpolation and Trajectory Tracking Part 2 of 5 Ellipse and Skewed-Astroid (x-y) parametric curves									
ITEM	DESCRIPTION	ELLIPSE CURVE					SKEWED-ASTROID CURVE				
		FC10	FC20	FC25	FC30	FC40	FC10	FC20	FC25	FC30	FC40
1	Run user feedrate command (mm/s)										
2	Total interpolated u-points	21575	7599	9448	8338	7351	116194	58102	46483	38738	29051
3	Parameter completion (reached u-end)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
<b>Pushdown epsilon eps(u) algorithm</b>											
4	Count before pushdown, eps(u) is below (1E-6)	21396	1427	7046	5149	928	116194	58102	46483	38738	29051
5	Count pushdown points, eps(u) to below (1E-6)	179	6172	2402	3189	6423	0	0	0	0	0
<b>Epsilon eps(u) chord error</b>											
6	Count eps(u) above (1E-6)	0	0	0	0	0	0	0	0	0	0
7	Count eps(u) in (1E-7, 1E-6)	7058	7599	9448	8338	7351	0	0	0	1455	843
8	Count eps(u) in (1E-8, 1E-7)	14517	0	0	0	0	9483	33004	46483	37283	2062
9	Count eps(u) in (1E-9, 1E-8)	0	0	0	0	0	106711	25098	0	0	0
10	Count eps(u) in (1E-10, 1E-9)	0	0	0	0	0	0	0	0	0	0
11	Count eps(u) below (1E-10)	0	0	0	0	0	0	0	0	0	0
<b>Count interpolated u-points</b>											
12	Count rising S curve u-points	443	480	370	370	370	3673	1837	1470	1225	914
13	Count frate is lower than fratelimit	14764	4342	6120	5270	3308	75069	39802	31837	26531	1993
14	Count frate is equal to fratelimit	0	0	0	0	0	0	0	0	0	0
15	Count frate is higher than fratelimit	5926	2298	2589	2329	3304	33779	14626	11707	9757	727
16	Count falling S curve u-points	442	479	369	369	369	3673	1837	1469	1225	914
<b>Count u-points histogram (G01 codes)</b>											
17	Count u-points [0.00 <= u < 0.10]	1231	779	748	748	748	7322	3661	2929	2441	183
18	Count u-points [0.10 <= u < 0.20]	2586	1292	1035	878	791	12603	6302	5042	4202	315
19	Count u-points [0.20 <= u < 0.30]	3154	1578	1262	1051	797	18245	9123	7298	6081	456
20	Count u-points [0.30 <= u < 0.40]	2586	1293	1034	863	710	12604	6302	5042	4202	315
21	Count u-points [0.40 <= u < 0.50]	1230	704	644	628	629	7322	3662	2930	2442	183
22	Count u-points [0.50 <= u < 0.60]	1228	704	644	629	629	7323	3662	2930	2442	183
23	Count u-points [0.60 <= u < 0.70]	2586	1293	1034	862	710	12603	6302	5041	4201	315
24	Count u-points [0.70 <= u < 0.80]	3155	1577	1262	1051	796	18244	9122	7298	6082	456
25	Count u-points [0.80 <= u < 0.90]	2586	1293	1035	878	791	12604	6303	5043	4202	315
26	Count u-points [0.90 <= u <= 1.00]	1233	782	750	750	750	7324	3663	2930	2443	183
27	Check Total u-points	21575	11295	9448	8338	7351	116194	58102	46483	38738	29051
<b>Performance</b>											
28	Total curve error (sum of epsilon(u))	0.002991	0.007141	0.005928	0.006562	0.007332	0.000516	0.001033	0.001291	0.001549	0.00204
29	Total dist traversed (sum of chord lengths)	215.6437	215.6499	215.644	215.6478	215.6439	445.7143	445.7143	445.7143	445.7143	445.714
30	Percent (Tot curve error / Tot dist traversed)	0.001387	0.003311	0.002749	0.003043	0.0034	0.000116	0.000232	0.00029	0.000347	0.00046

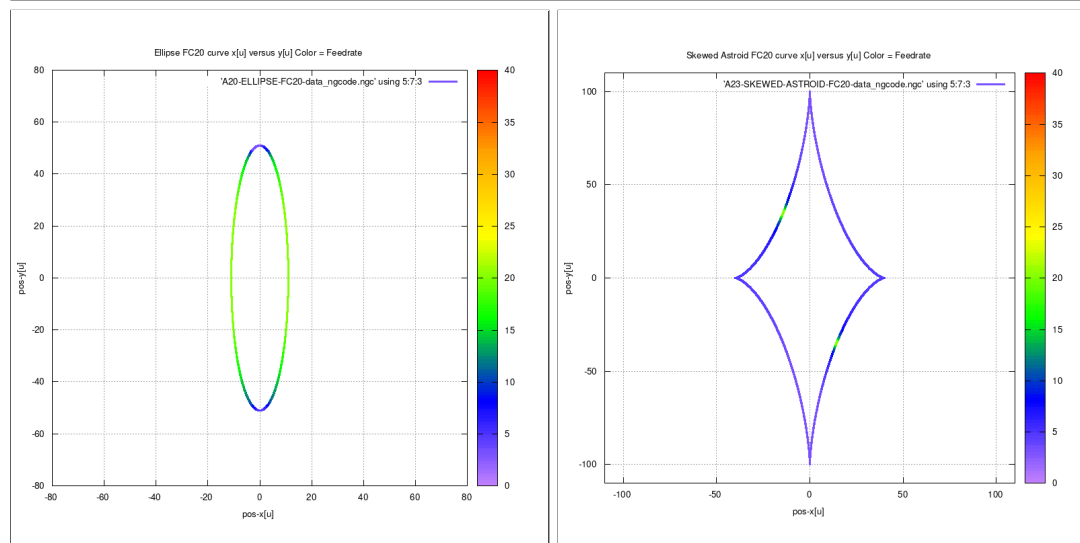


Table 5.22: Ellipse and Skewed-Astroid Run Data

### 5.3.3 Circle and Astepi Run Data

Date: 2023-06-06 Author: wruslandr@gmail.com		Report CNC Parametric Curve Interpolation and Trajectory Tracking Part 3 of 5 Circle and Astepi (x-y) parametric curves									
ITEM	DESCRIPTION	CIRCLE CURVE					ASTEROID + EPICYCLOID CURVE				
		FC10	FC20	FC25	FC30	FC40	FC10	FC20	FC25	FC30	FC40
1	Run user feedrate command (mm/s)										
2	Total interpolated u-points	49641	24822	19859	16549	12413	76275	38169	30563	25499	19184
3	Parameter completion (reached u-end)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
<b>Pushdown epsilon eps(u) algorithm</b>											
4	Count before pushdown, eps(u) is below (1E-6)	49641	24822	19859	16549	12413	76275	38061	30413	25309	18917
5	Count pushdown points, eps(u) to below (1E-6)	0	0	0	0	0	0	108	150	190	267
<b>Epsilon eps(u) chord error</b>											
6	Count eps(u) above (1E-6)	0	0	0	0	0	0	0	0	0	0
7	Count eps(u) in (1E-7, 1E-6)	0	6959	17603	16549	12413	407	820	4011	6679	13976
8	Count eps(u) in (1E-8, 1E-7)	49641	17863	2256	0	0	23373	37349	26552	18820	5208
9	Count eps(u) in (1E-9, 1E-8)	0	0	0	0	0	52495	0	0	0	0
10	Count eps(u) in (1E-10, 1E-9)	0	0	0	0	0	0	0	0	0	0
11	Count eps(u) below (1E-10)	0	0	0	0	0	0	0	0	0	0
<b>Count interpolated u-points</b>											
12	Count rising S curve u-points	2483	1242	993	828	621	4437	2219	1775	1479	1110
13	Count frate is lower than fratelimit	22363	11171	8936	7449	5586	33349	16686	13352	11144	8085
14	Count frate is equal to fratelimit	0	0	0	0	0	0	0	0	0	0
15	Count frate is higher than fratelimit	22313	11168	8937	7445	5585	35536	17787	14254	11891	9250
16	Count falling S curve u-points	2482	1241	993	827	621	2953	1477	1182	985	739
<b>Count u-points histogram (G01 codes)</b>											
17	Count u-points [0.00 <= u < 0.10]	4964	2482	1985	1654	1241	8110	4055	3244	2704	2028
18	Count u-points [0.10 <= u < 0.20]	4964	2482	1986	1655	1241	4901	2478	2009	1702	1334
19	Count u-points [0.20 <= u < 0.30]	4964	2482	1986	1655	1241	7391	3696	2957	2464	1848
20	Count u-points [0.30 <= u < 0.40]	4964	2482	1986	1655	1241	7234	3617	2894	2412	1809
21	Count u-points [0.40 <= u < 0.50]	4964	2482	1986	1655	1242	9182	4592	3673	3061	2297
22	Count u-points [0.50 <= u < 0.60]	4964	2483	1985	1655	1241	7216	3608	2887	2406	1804
23	Count u-points [0.60 <= u < 0.70]	4964	2482	1986	1655	1241	9831	4916	3933	3278	2458
24	Count u-points [0.70 <= u < 0.80]	4964	2482	1986	1655	1241	7525	3763	3010	2508	1882
25	Count u-points [0.80 <= u < 0.90]	4964	2482	1986	1654	1242	8801	4401	3521	2935	2201
26	Count u-points [0.90 <= u <= 1.00]	4965	2483	1987	1656	1242	6084	3043	2435	2029	1523
27	Check Total u-points	49641	24822	19859	16549	12413	76275	38169	30563	25499	19184
<b>Performance</b>											
28	Total curve error (sum of epsilon(u))	0.001094	0.001094	0.002188	0.002735	0.004375	0.00084	0.001642	0.00202	0.00239	0.003111
29	Total dist traversed (sum of chord lengths)	496.3786	496.3772	496.3964	496.3757	496.3943	426.2622	426.2622	426.2622	426.2622	426.2622
30	Percent (Tot curve error / Tot dist traversed)	0.00022	0.00022	0.000441	0.000551	0.000881	0.000197	0.000385	0.000474	0.000561	0.00073

Table 5.23: Circle and Astepi Run Data

### 5.3.4 Snailshell and SnaHyp Run Data

Date: 2023-06-06 Author: wruslandr@gmail.com		Report CNC Parametric Curve Interpolation and Trajectory Tracking Part 4 of 5 Snailshell and SnaHyp (x-y) parametric curves									
ITEM	DESCRIPTION	SNAILSHELL CURVE					SNAILSHELL + HYPOTROCID CURVE				
		FC10	FC20	FC25	FC30	FC40	FC10	FC20	FC25	FC30	FC40
1	Run user feedrate command (mm/s)										
2	Total interpolated u-points	15621	9883	8935	8370	7766	38672	20223	16618	11497	8889
3	Parameter completion (reached u-end = 1.00)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.706104	0.703536
<b>Pushdown epsilon(u) algorithm</b>											
4	Count before pushdown, eps(u) is below (1E-6)	12245	5010	3569	2671	1592	37227	18167	14266	0	0
5	Count pushdown points, eps(u) to below (1E-6)	3376	4873	5366	5699	6174	1445	2056	2352	0	0
<b>Epsilon eps(u) chord error</b>											
6	Count eps(u) above (1E-6)	0	0	0	0	0	0	0	0	0	0
7	Count eps(u) in (1E-7, 1E-6)	7893	9323	8935	8370	7766	4106	6971	9900	0	0
8	Count eps(u) in (1E-8, 1E-7)	7728	560	0	0	0	24178	13252	6718	0	0
9	Count eps(u) in (1E-9, 1E-8)	0	0	0	0	0	10388	0	0	0	0
10	Count eps(u) in (1E-10, 1E-9)	0	0	0	0	0	0	0	0	0	0
11	Count eps(u) below (1E-10)	0	0	0	0	0	0	0	0	0	0
<b>Count interpolated u-points</b>											
12	Count rising S curve u-points	2320	1161	929	774	581	3177	1589	1272	0	0
13	Count frate is lower than fratelimit	11722	6605	5826	5386	4860	18305	9773	8142	0	0
14	Count frate is equal to fratelimit	0	0	0	0	0	0	0	0	0	0
15	Count frate is higher than fratelimit	1210	1747	1811	1859	1956	14859	7695	6271	0	0
16	Count falling S curve u-points	369	370	359	369	369	2331	1166	933	0	0
<b>Count u-points histogram (G01 codes)</b>											
17	Count u-points [0.00 <= u < 0.10]	4435	2218	1774	1479	1109	4631	2317	1856	1563	1217
18	Count u-points [0.10 <= u < 0.20]	3237	1619	1296	1080	849	8961	4480	3584	2987	2240
19	Count u-points [0.20 <= u < 0.30]	2054	1028	851	796	793	6140	3074	2470	2072	1586
20	Count u-points [0.30 <= u < 0.40]	1312	714	710	711	711	2960	1526	1257	1086	885
21	Count u-points [0.40 <= u < 0.50]	881	629	629	629	629	4860	2431	1945	1620	1216
22	Count u-points [0.50 <= u < 0.60]	657	628	628	629	628	3973	1987	1589	1325	994
23	Count u-points [0.60 <= u < 0.70]	710	711	711	710	711	1324	841	769	732	710
24	Count u-points [0.70 <= u < 0.80]	794	794	794	794	794	794	794	794	112	41
25	Count u-points [0.80 <= u < 0.90]	791	791	792	792	792	1141	828	798	0	0
26	Count u-points [0.90 <= u <= 1.00]	750	751	750	750	750	3888	1945	1556	0	0
27	Check Total u-points	15621	9883	8935	8370	7766	38672	20223	16618	11497	8889
<b>Performance</b>											
28	Total curve error (sum of epsilon(u))	0.005115	0.00627	0.006558	0.006764	0.007046	0.002847	0.004003	0.004459	0	0
29	Total dist traversed (sum of chord lengths)	138.5595	138.5614	138.5607	138.5602	138.5599	478.9871	478.9987	479.0064	0	0
30	Percent (Tot curve error / Tot dist traversed)	0.003692	0.004525	0.004733	0.004882	0.005085	0.000594	0.000836	0.000931	0	0

Table 5.24: Snailshell and SnaHyp Run Data

### 5.3.5 Ribbon-10L and Ribbon-100L Run Data

Date: 2023-06-06 Author: <a href="mailto:wruslandr@gmail.com">wruslandr@gmail.com</a>		Report CNC Parametric Curve Interpolation and Trajectory Tracking Part 5 of 5 Ribbon-10L and Ribbon-100L (x-y) parametric curves									
ITEM	DESCRIPTION	RIBBON-10L CURVE					RIBBON-100L CURVE				
		FC10	FC20	FC25	FC30	FC40	FC10	FC20	FC25	FC30	FC40
1	Run user feedrate command (mm/s)										
2	Total interpolated u-points	7351	7352	7352	7353	7353	7480	7348	7349	7349	7350
3	Parameter completion (reached u-end)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
<b>Pushdown epsilon eps(u) algorithm</b>											
4	Count before pushdown, eps(u) is below (1E-6)	0	0	0	0	0	919	0	0	0	0
5	Count pushdown points, eps(u) to below (1E-6)	7351	7352	7352	7353	7353	6561	7348	7349	7349	7350
<b>Epsilon eps(u) chord error</b>											
6	Count eps(u) above (1E-6)	0	0	0	0	0	0	0	0	0	0
7	Count eps(u) in (1E-7, 1E-6)	7351	7352	7352	7353	7353	7480	7348	7349	7349	7350
8	Count eps(u) in (1E-8, 1E-7)	0	0	0	0	0	0	0	0	0	0
9	Count eps(u) in (1E-9, 1E-8)	0	0	0	0	0	0	0	0	0	0
10	Count eps(u) in (1E-10, 1E-9)	0	0	0	0	0	0	0	0	0	0
11	Count eps(u) below (1E-10)	0	0	0	0	0	0	0	0	0	0
<b>Count interpolated u-points</b>											
12	Count rising S curve u-points	370	370	370	370	370	436	370	370	370	370
13	Count frate is lower than fratelimit	3491	3310	3310	3308	3309	5562	3939	3733	3545	3307
14	Count frate is equal to fratelimit	0	0	0	0	0	0	0	0	0	0
15	Count frate is higher than fratelimit	3121	3303	3303	3305	3305	1047	2670	2877	3065	3303
16	Count falling S curve u-points	369	369	369	370	369	435	369	369	369	370
<b>Count u-points histogram (G01 codes)</b>											
17	Count u-points [0.00 <= u < 0.10]	749	749	749	749	749	815	748	748	748	749
18	Count u-points [0.10 <= u < 0.20]	791	792	792	792	792	791	792	792	792	791
19	Count u-points [0.20 <= u < 0.30]	795	794	794	794	794	794	794	794	794	794
20	Count u-points [0.30 <= u < 0.40]	711	711	711	711	712	711	711	711	711	711
21	Count u-points [0.40 <= u < 0.50]	629	629	629	630	629	629	628	629	629	629
22	Count u-points [0.50 <= u < 0.60]	629	630	630	629	629	628	629	628	629	629
23	Count u-points [0.60 <= u < 0.70]	711	711	711	711	711	711	711	712	711	711
24	Count u-points [0.70 <= u < 0.80]	794	794	794	795	795	794	794	794	794	794
25	Count u-points [0.80 <= u < 0.90]	792	792	792	791	792	791	791	791	791	792
26	Count u-points [0.90 <= u <= 1.00]	750	750	750	751	750	816	750	750	750	750
27	Check Total u-points	7351	7352	7352	7353	7353	7480	7348	7349	7349	7350
<b>Performance</b>											
28	Total curve error (sum of epsilon(u))	0.007332	0.007331	0.007333	0.007331	0.007333	0.007227	0.007334	0.007335	0.007334	0.007334
29	Total dist traversed (sum of chord lengths)	15.2108	15.21069	15.20945	15.21391	15.21192	152.0974	152.1029	152.1321	152.1103	152.1394
30	Percent (Tot curve error / Tot dist traversed)	0.048201	0.048194	0.048195	0.048185	0.048187	0.004752	0.004822	0.004821	0.004821	0.00482

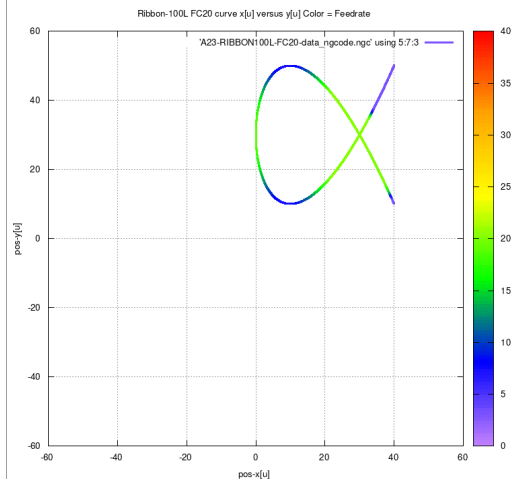
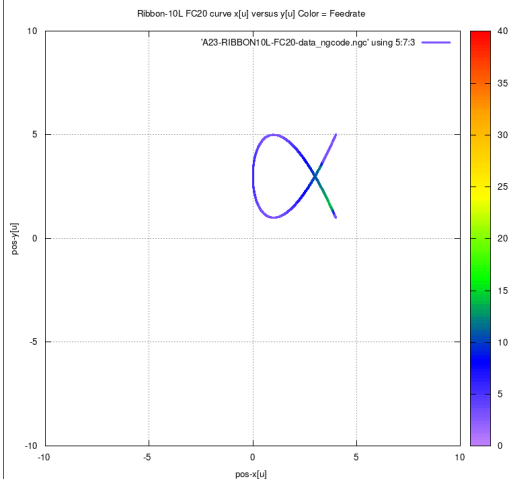


Table 5.25: Ribbon-10L and Ribbon-100L Run Data

## 5.4 Results Feedrate Profile

### 5.4.1 Teardrop FC20 u versus x-y-curr feedrate profile

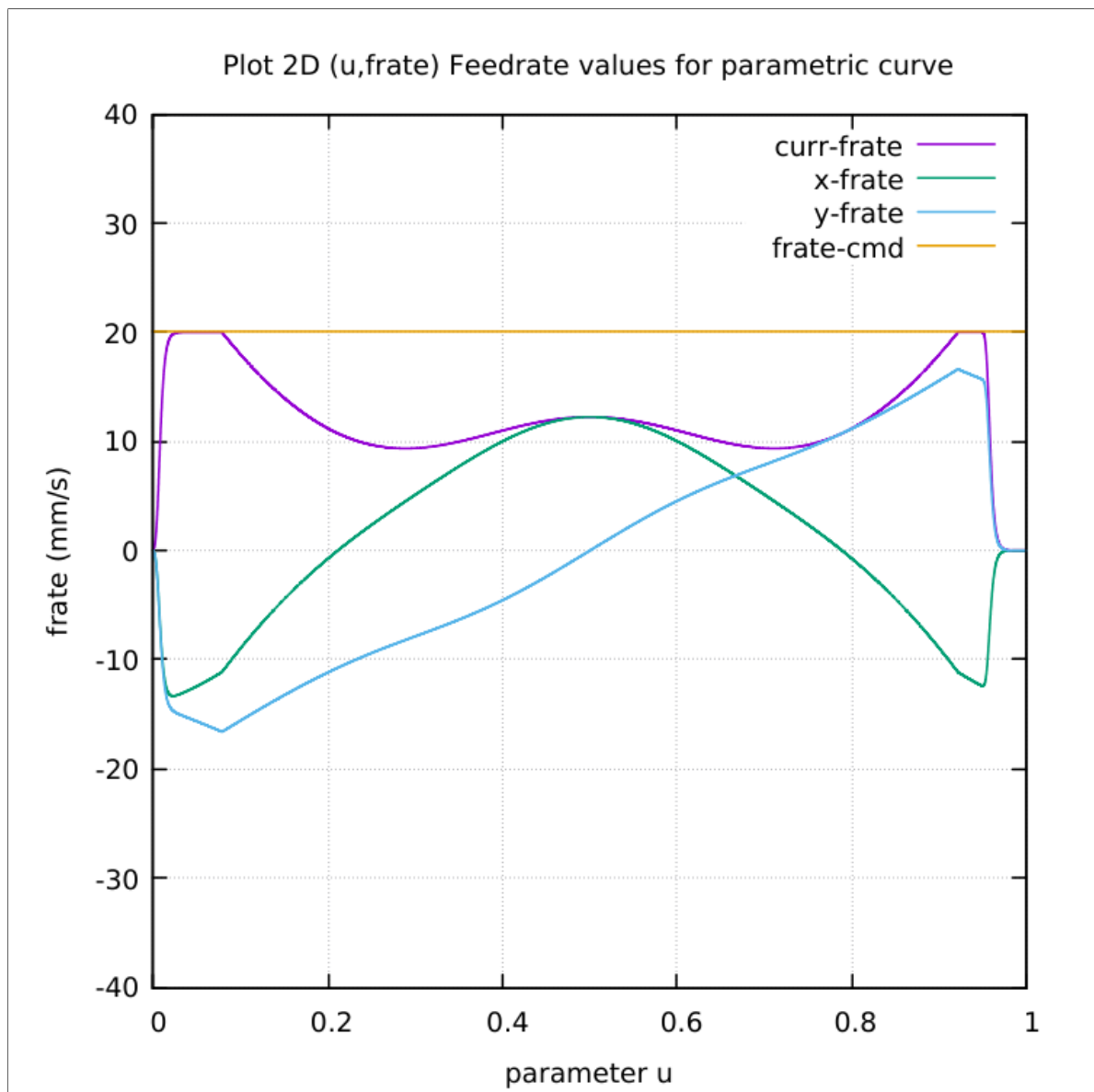


Table 5.26: Teardrop FC20 u versus x-y-curr feedrate profile

### 5.4.2 Teardrop FC20 x-y and colored feedrate profile

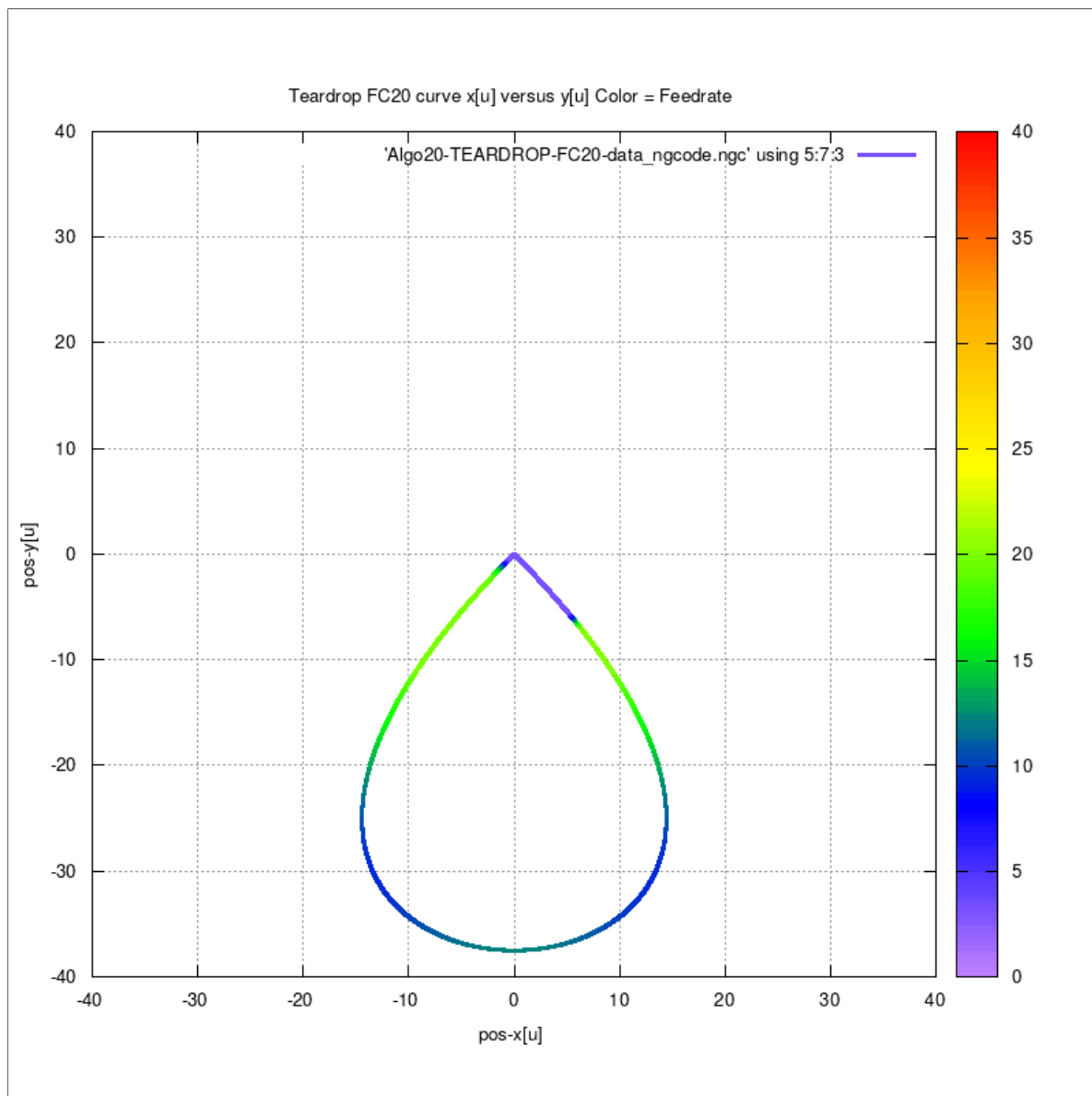


Table 5.27: Teardrop FC20 x-y and colored feedrate profile

### 5.4.3 Butterfly FC20 u versus x-y-curr feedrate profile

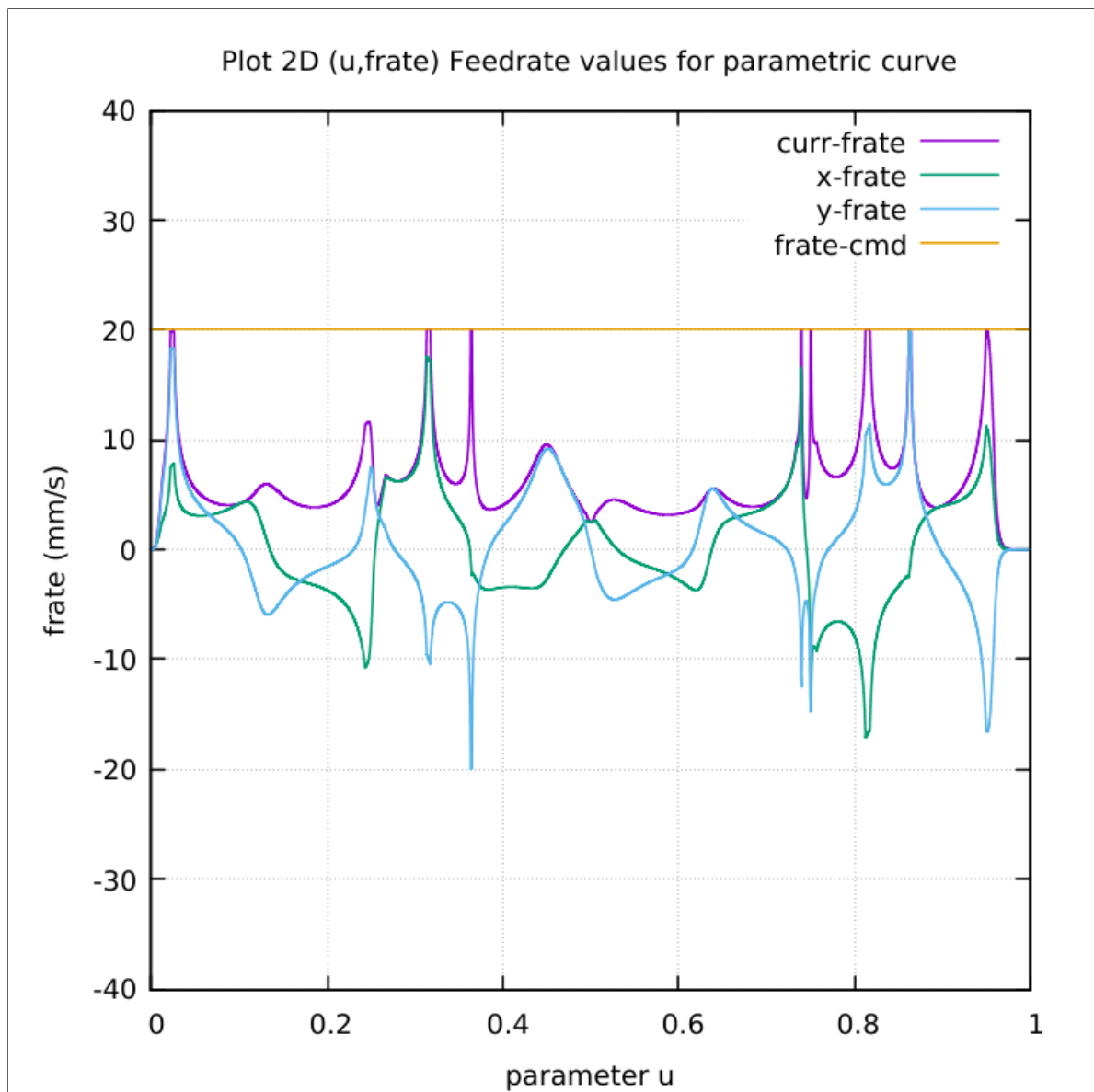


Table 5.28: Butterfly FC20 u versus x-y-curr feedrate profile



#### 5.4.4 Butterfly FC20 x-y and colored feedrate profile

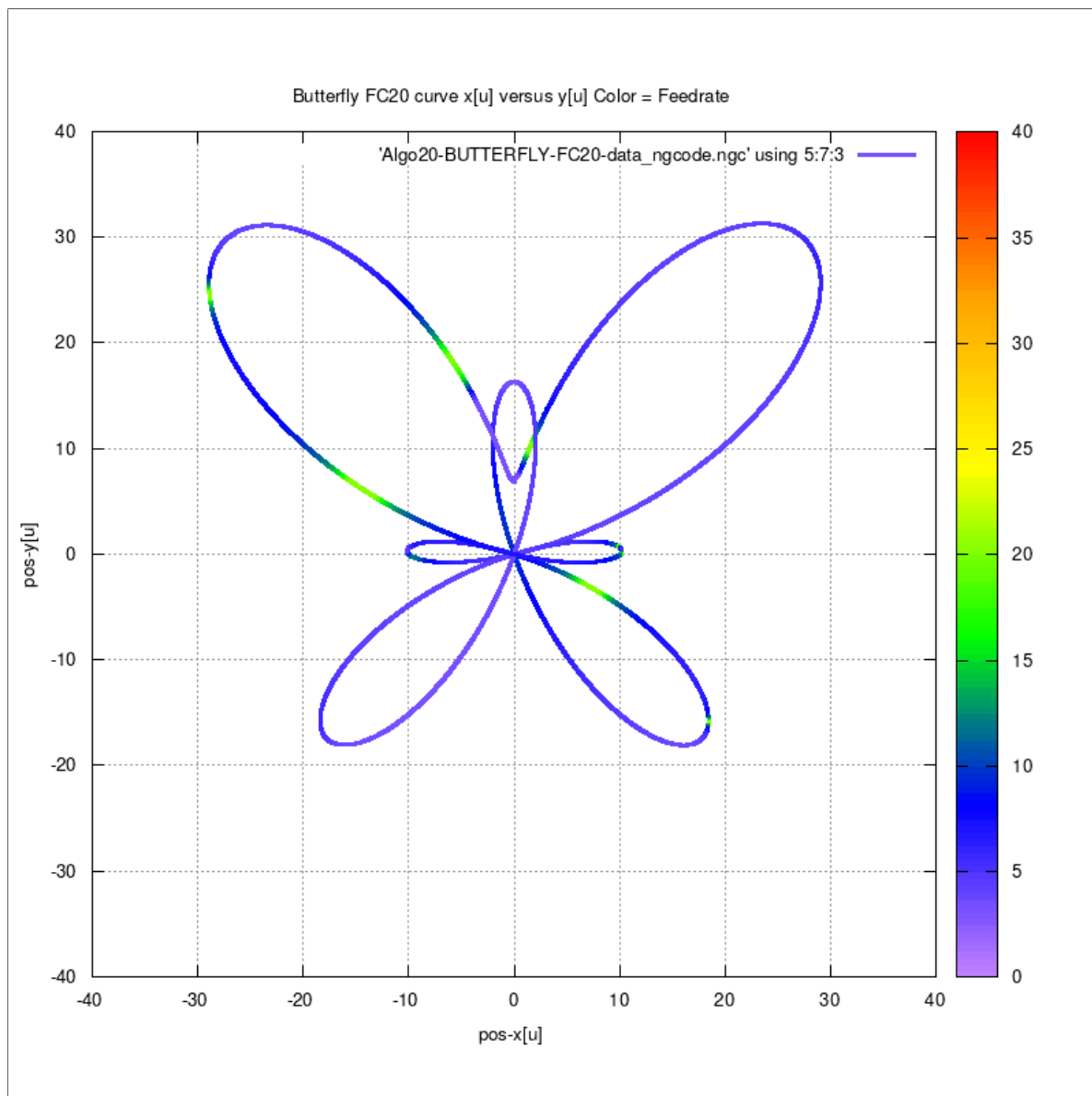


Table 5.29: Butterfly FC20 x-y and colored feedrate profile

### 5.4.5 Ellipse FC20 u versus x-y-curr feedrate profile

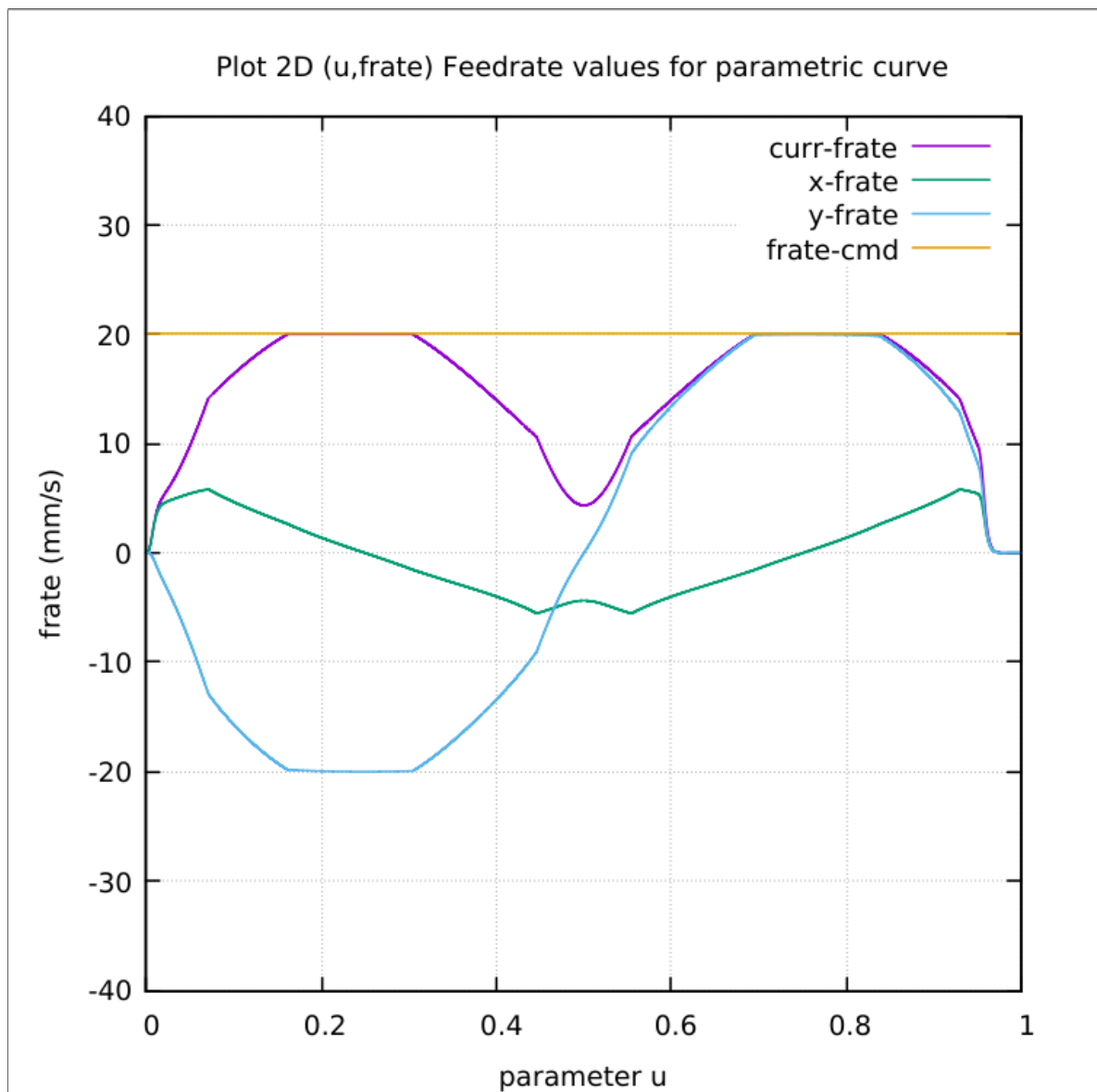


Table 5.30: Ellipse FC20 u versus x-y-curr feedrate profile

### 5.4.6 Ellipse FC20 x-y and colored feedrate profile

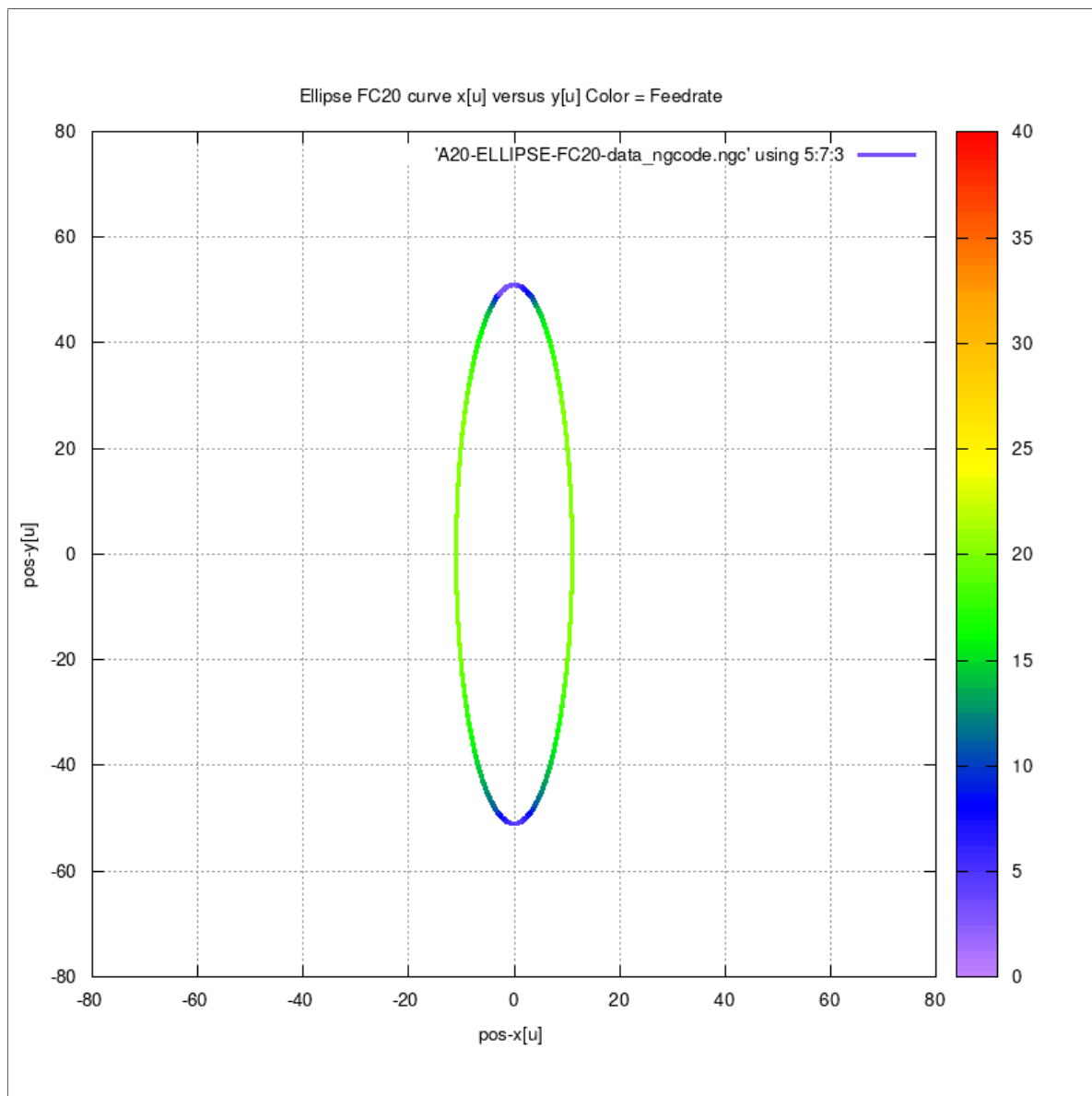


Table 5.31: Ellipse FC20 x-y and colored feedrate profile

### 5.4.7 Skewed-Astroid FC20 u versus x-y-curr feedrate profile

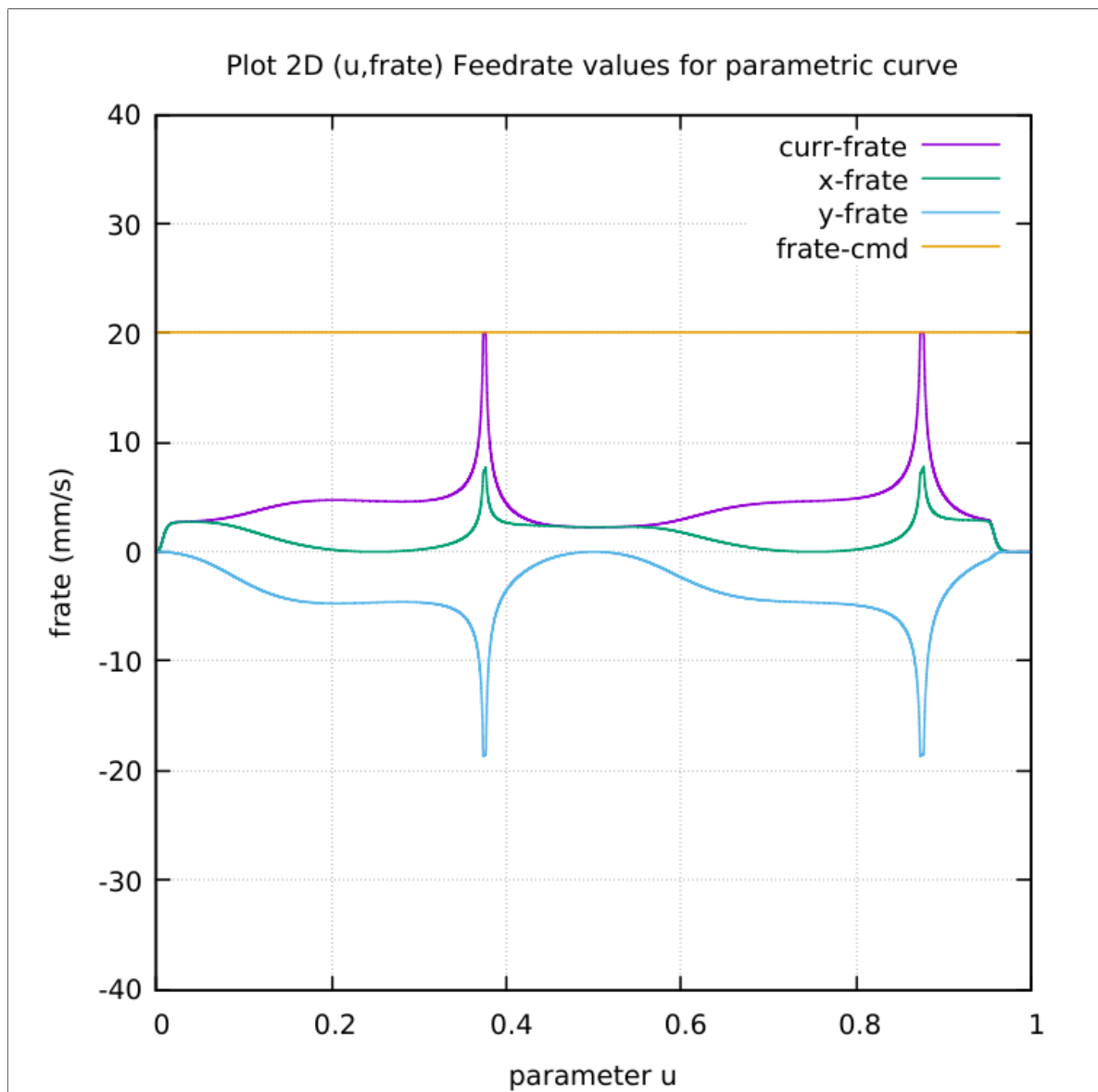


Table 5.32: Skewed-Astroid FC20 u versus x-y-curr feedrate profile

### 5.4.8 Skewed-Astroid FC20 x-y and colored feedrate profile

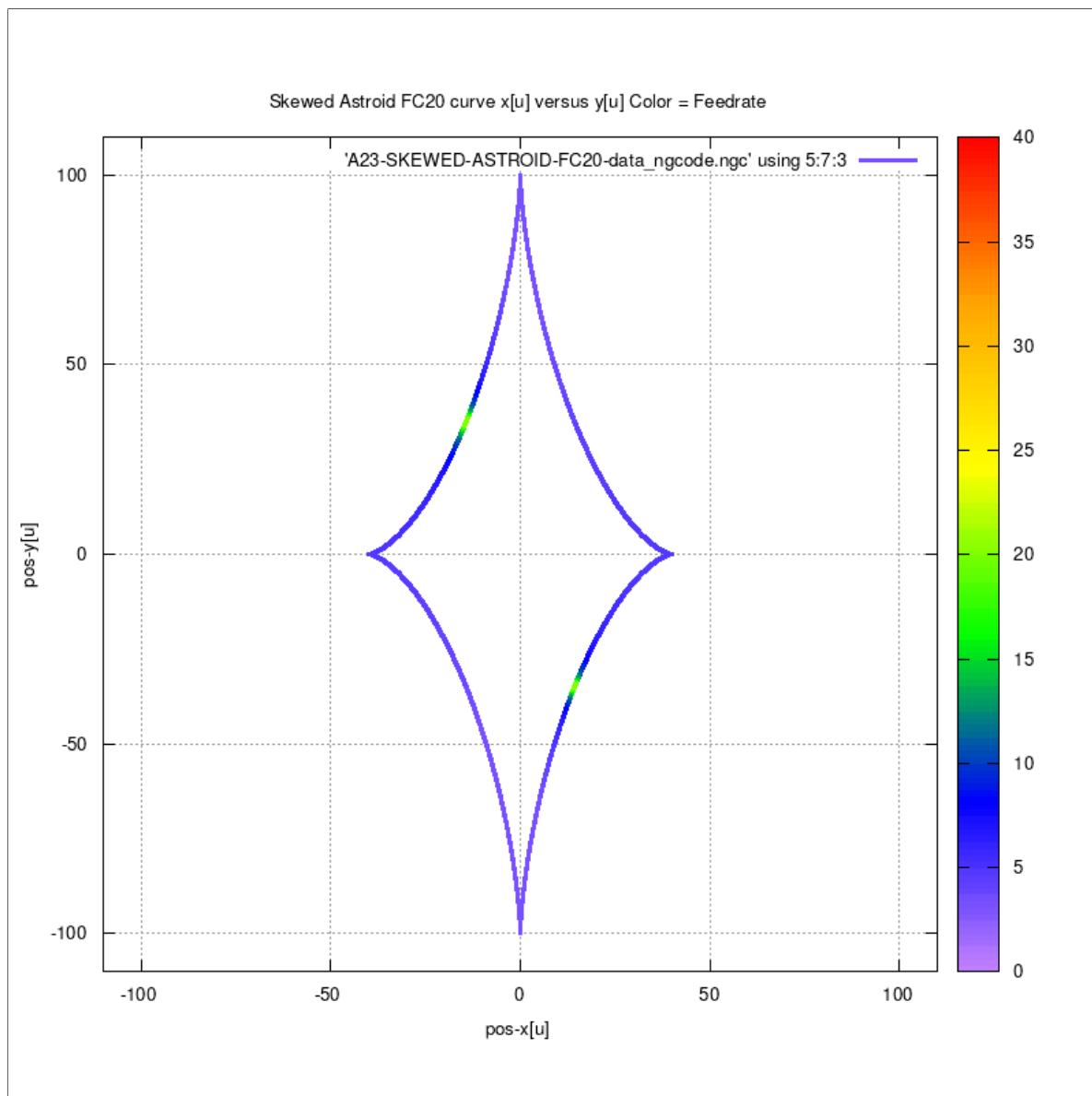


Table 5.33: Skewed-Astroid FC20 x-y and colored feedrate profile

### 5.4.9 Circle FC20 u versus x-y-curr feedrate profile

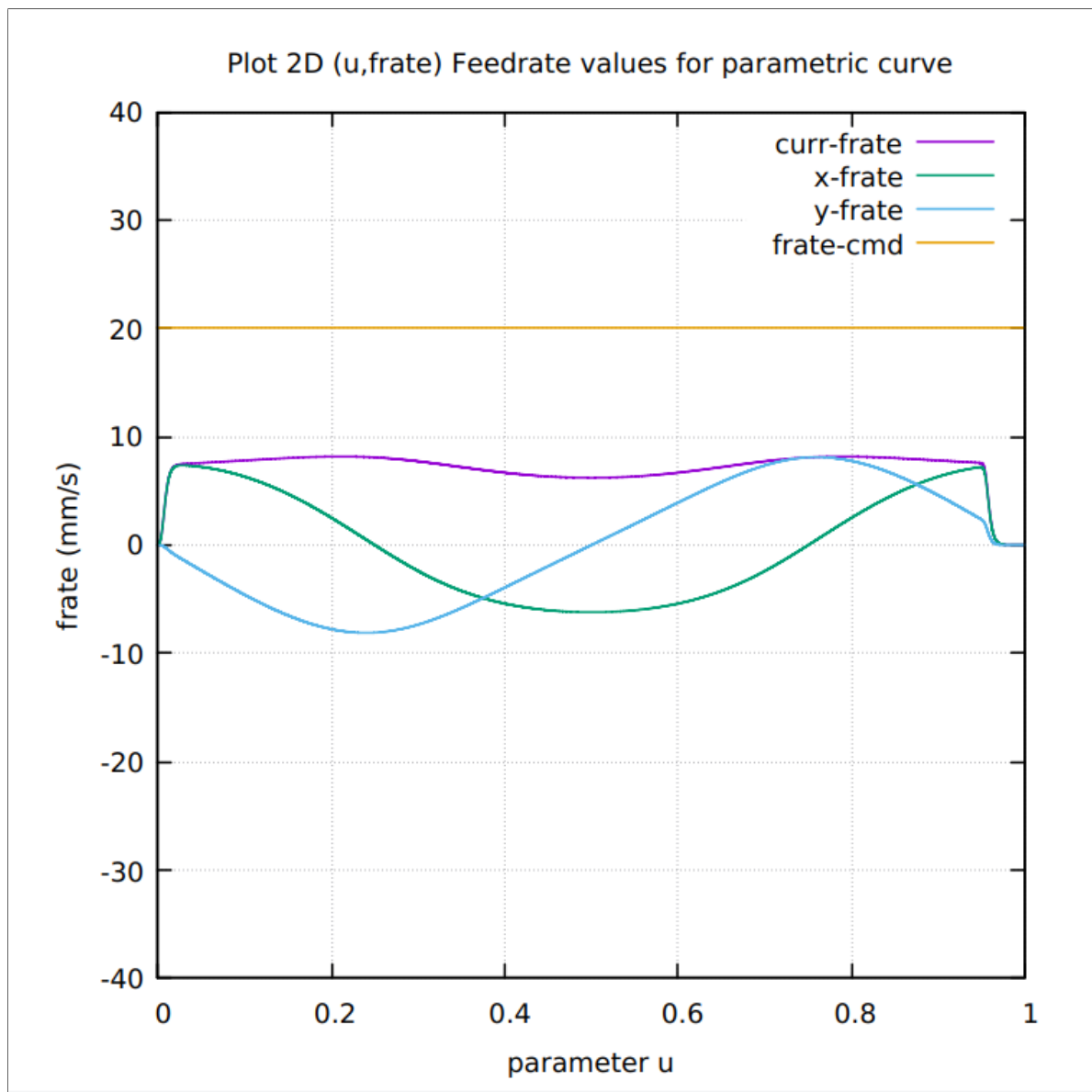


Table 5.34: Circle FC20 u versus x-y-curr feedrate profile

### 5.4.10 Circle FC20 x-y and colored feedrate profile

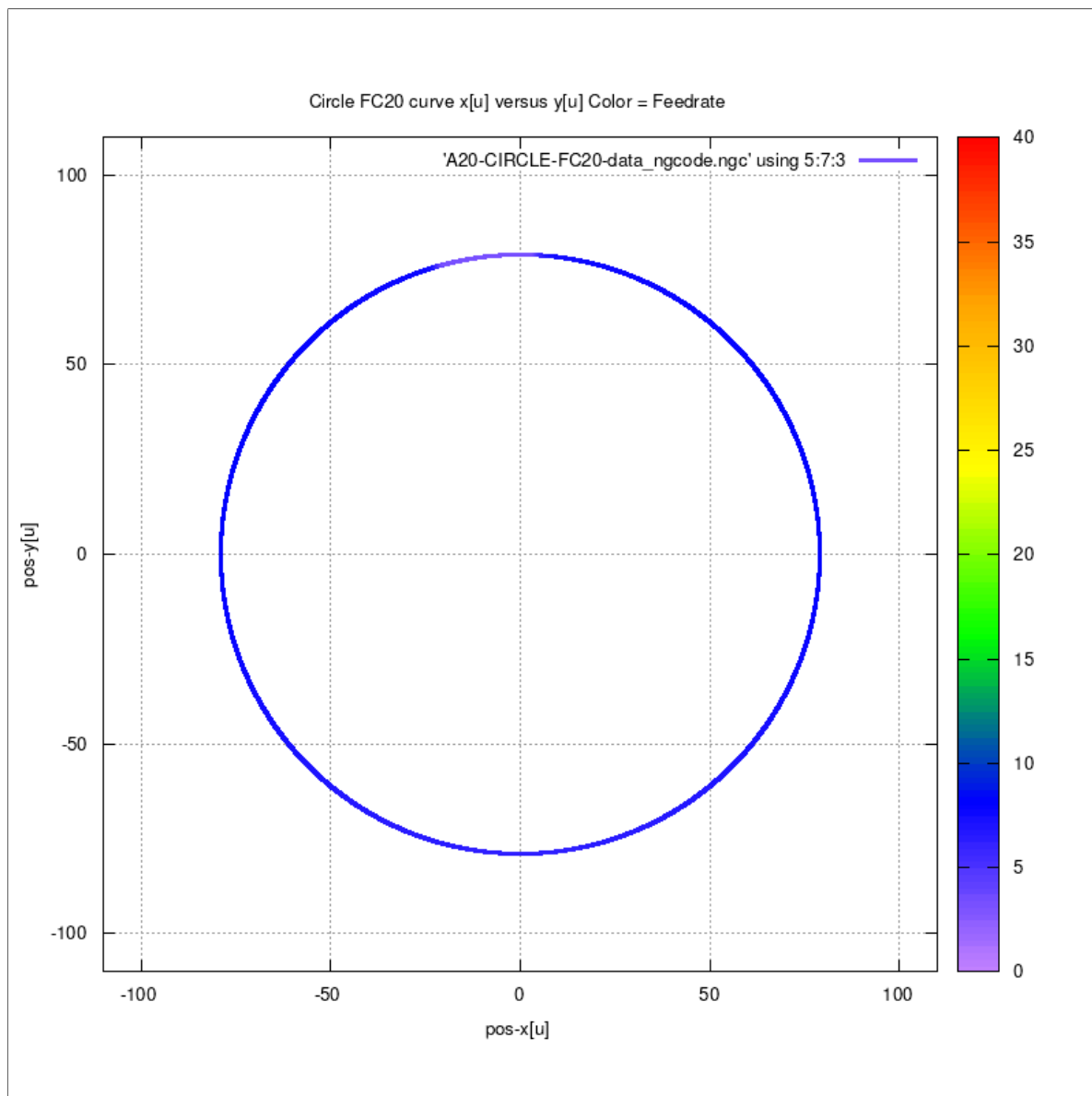


Table 5.35: Circle FC20 x-y and colored feedrate profile

### 5.4.11 AstEpi FC20 u versus x-y-curr feedrate profile

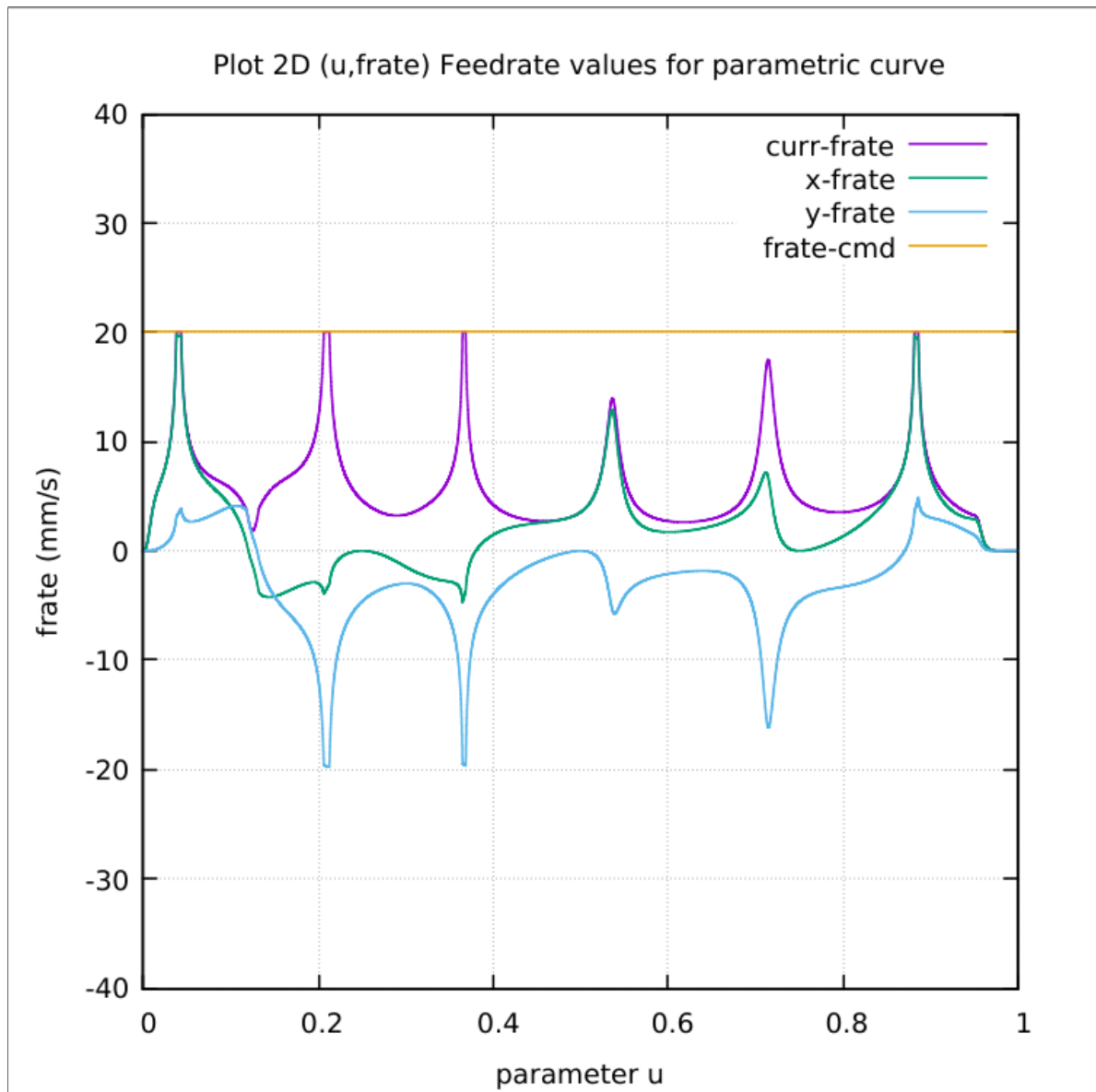


Table 5.36: AstEpi FC20 u versus x-y-curr feedrate profile



### 5.4.12 AstEpi FC20 x-y and colored feedrate profile

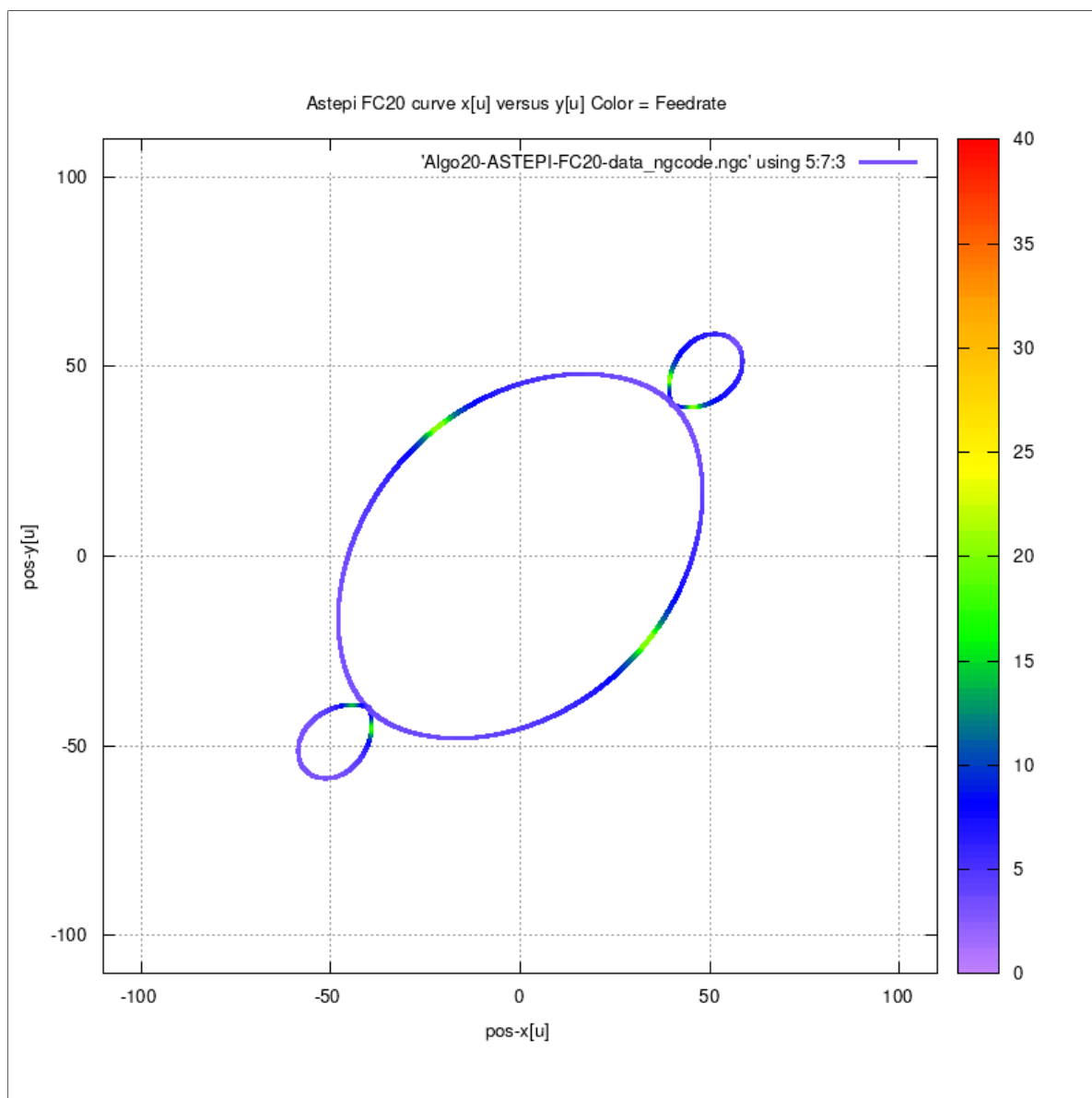


Table 5.37: AstEpi FC20 x-y and colored feedrate profile

### 5.4.13 Snailshell FC20 u versus x-y-curr feedrate profile

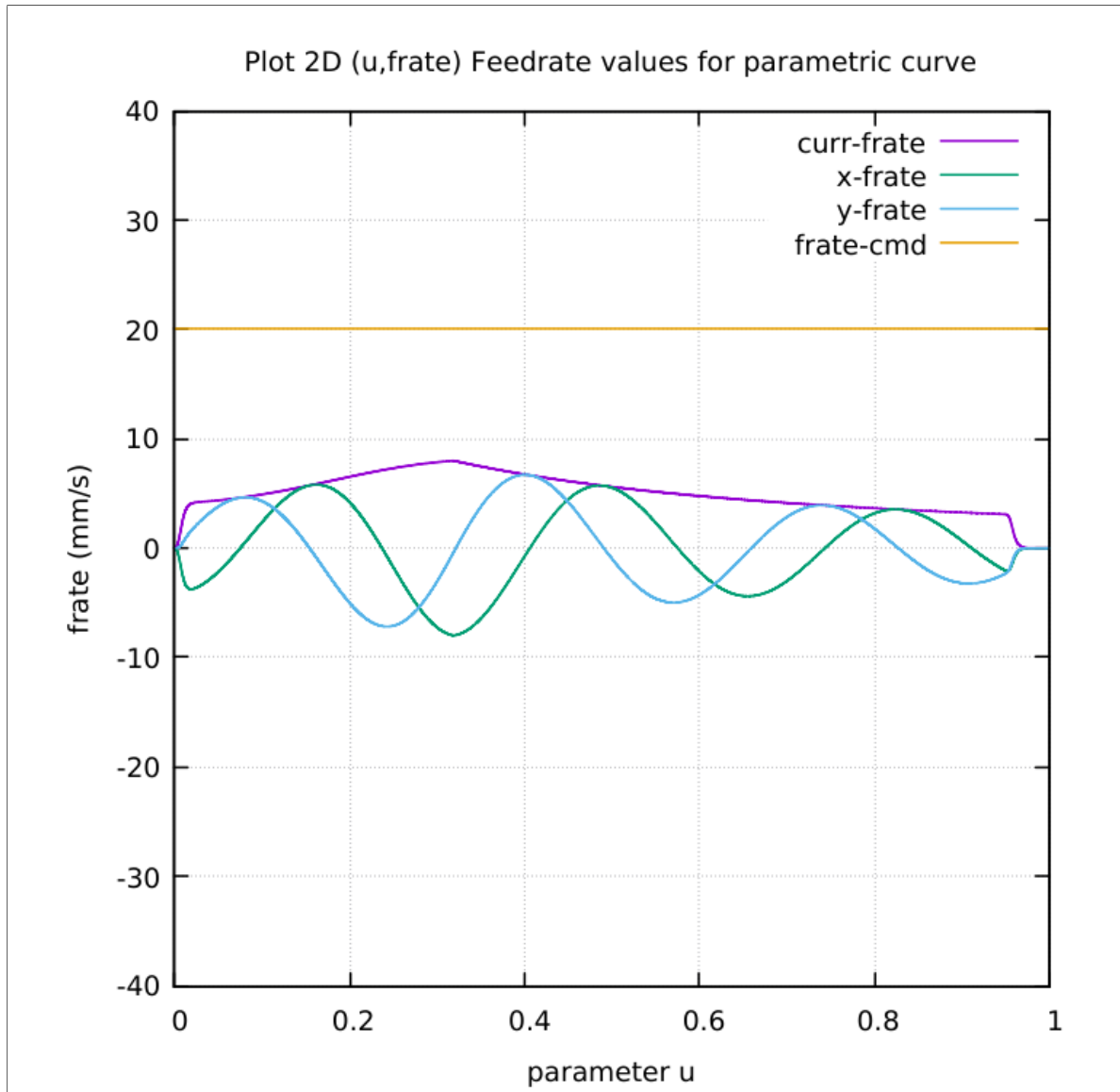


Table 5.38: Snailshell FC20 u versus x-y-curr feedrate profile

#### 5.4.14 Snailshell FC20 x-y and colored feedrate profile

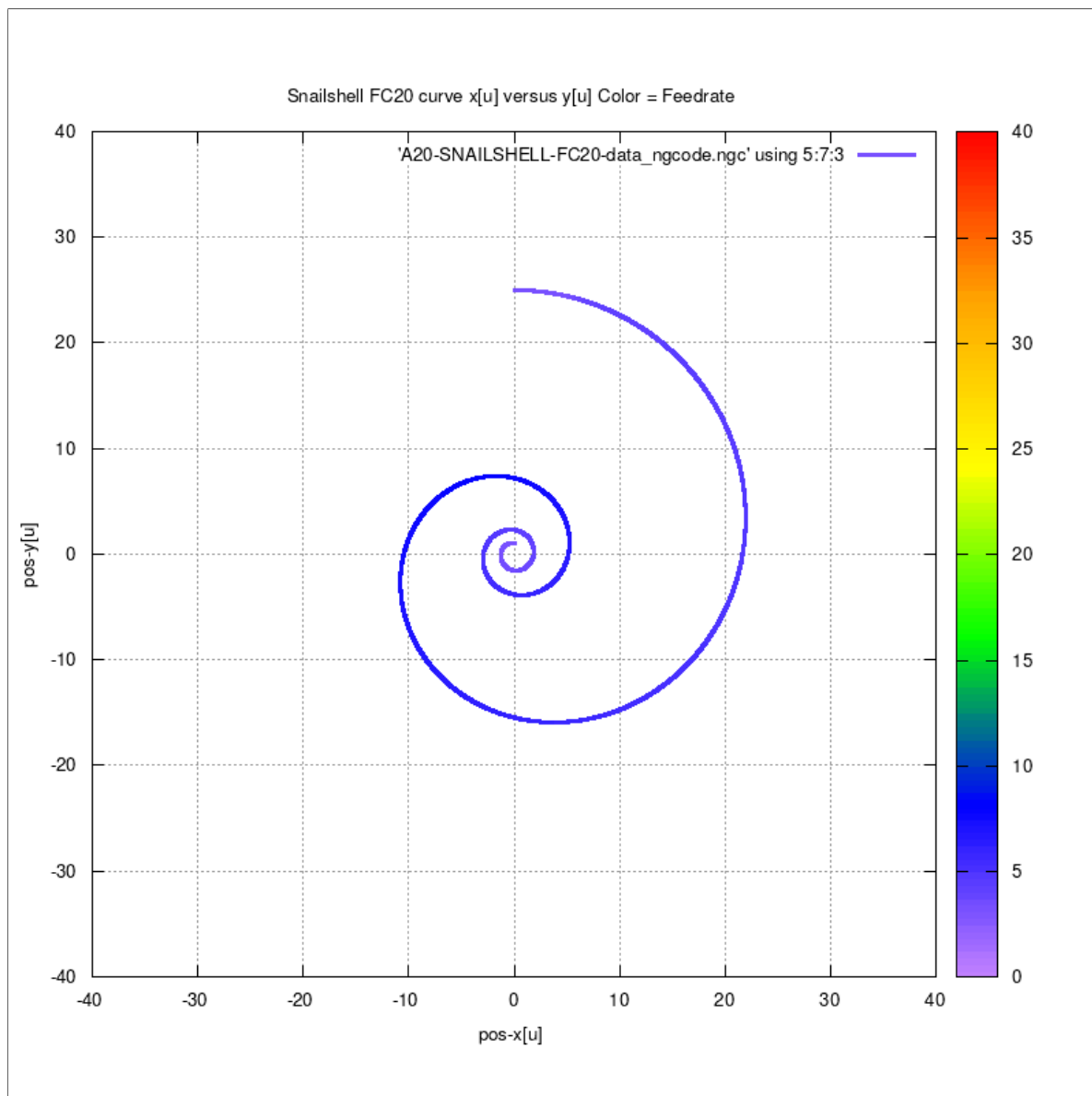


Table 5.39: Snailshell FC20 x-y and colored feedrate profile

### 5.4.15 SnaHyp FC20 u versus x-y-curr feedrate profile

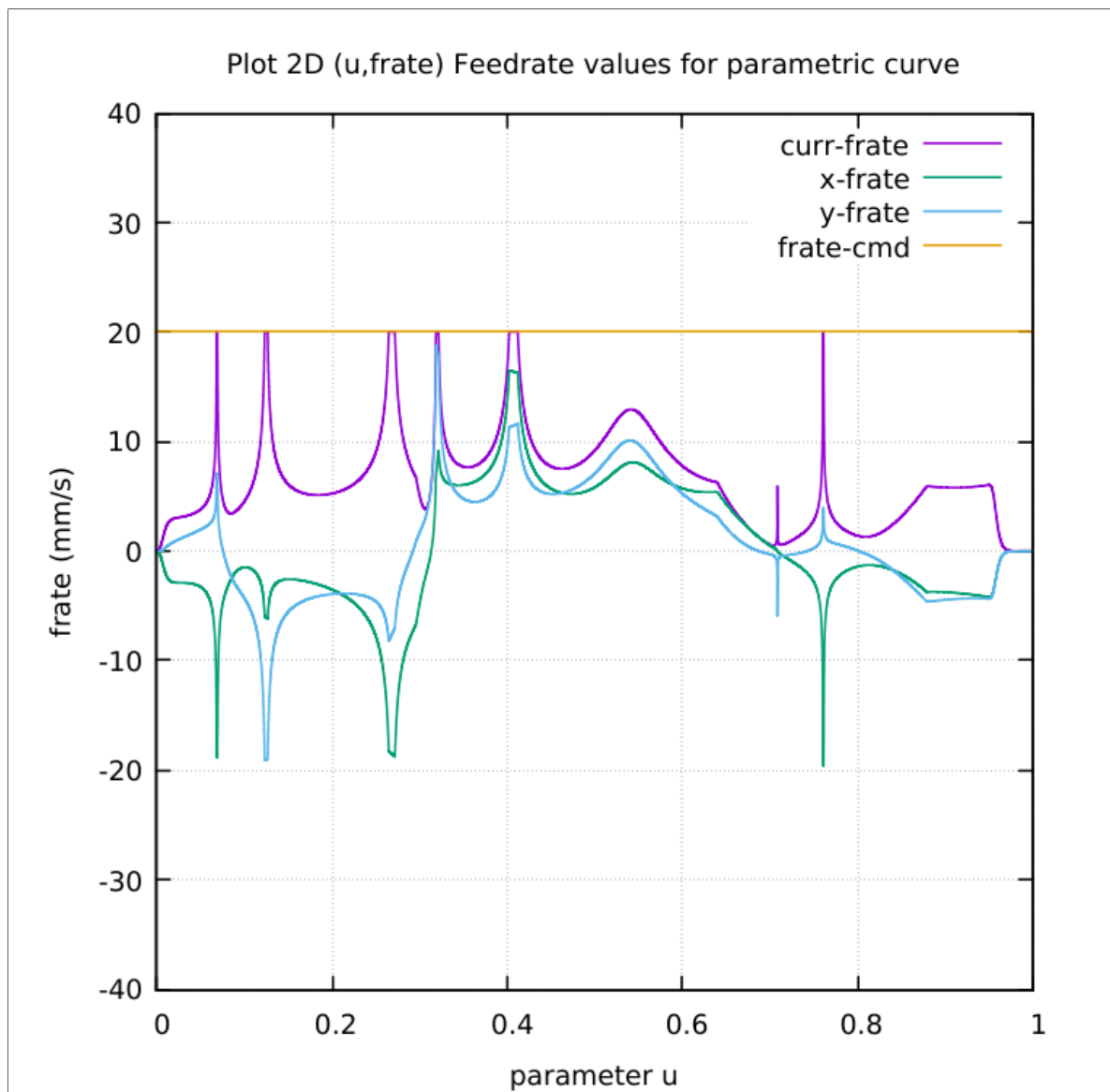


Table 5.40: SnaHyp FC20 u versus x-y-curr feedrate profile

### 5.4.16 SnaHyp FC20 x-y and colored feedrate profile

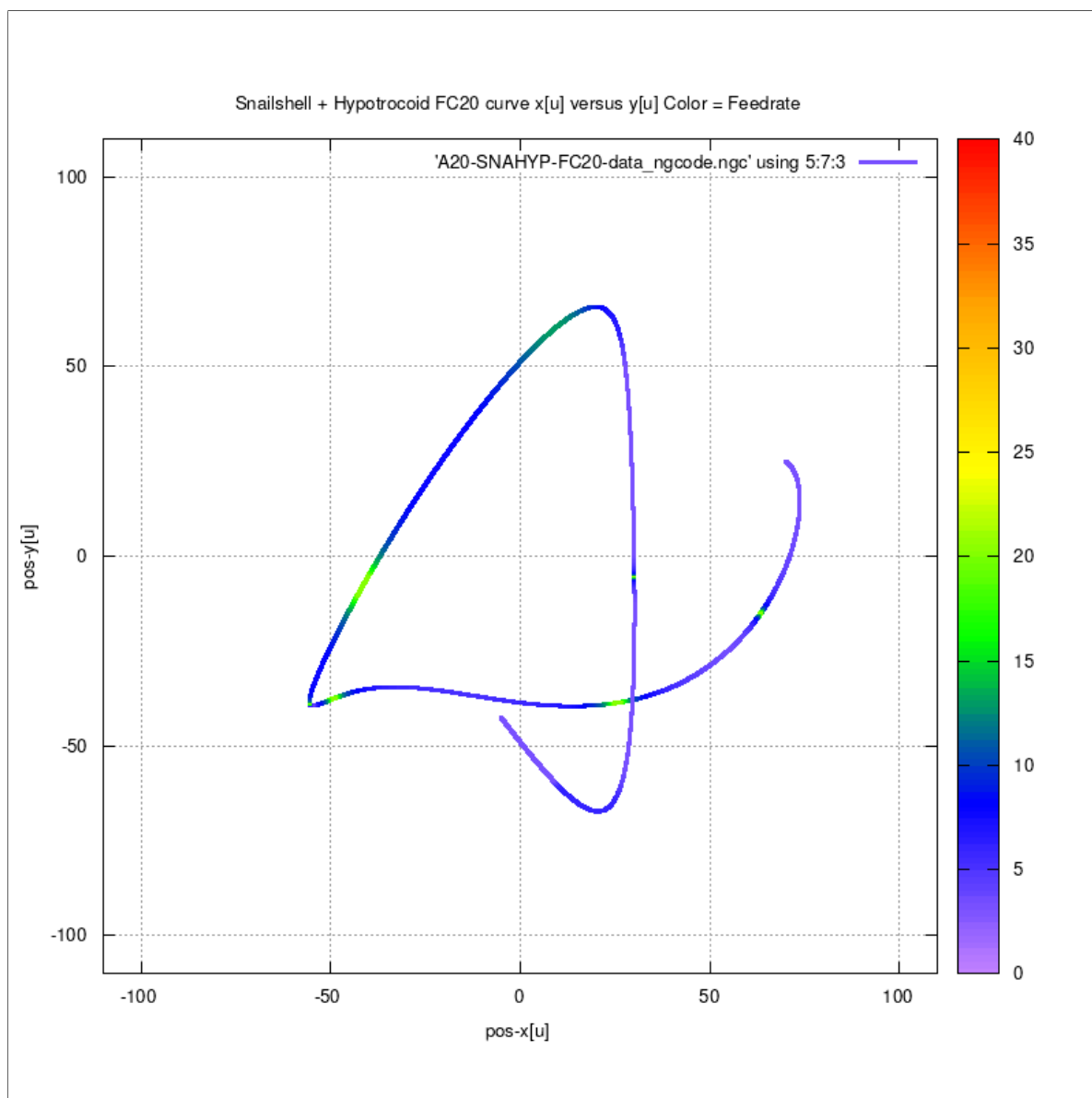


Table 5.41: SnaHyp FC20 x-y and colored feedrate profile

### 5.4.17 Ribbon-10L FC20 u versus x-y-curr feedrate profile

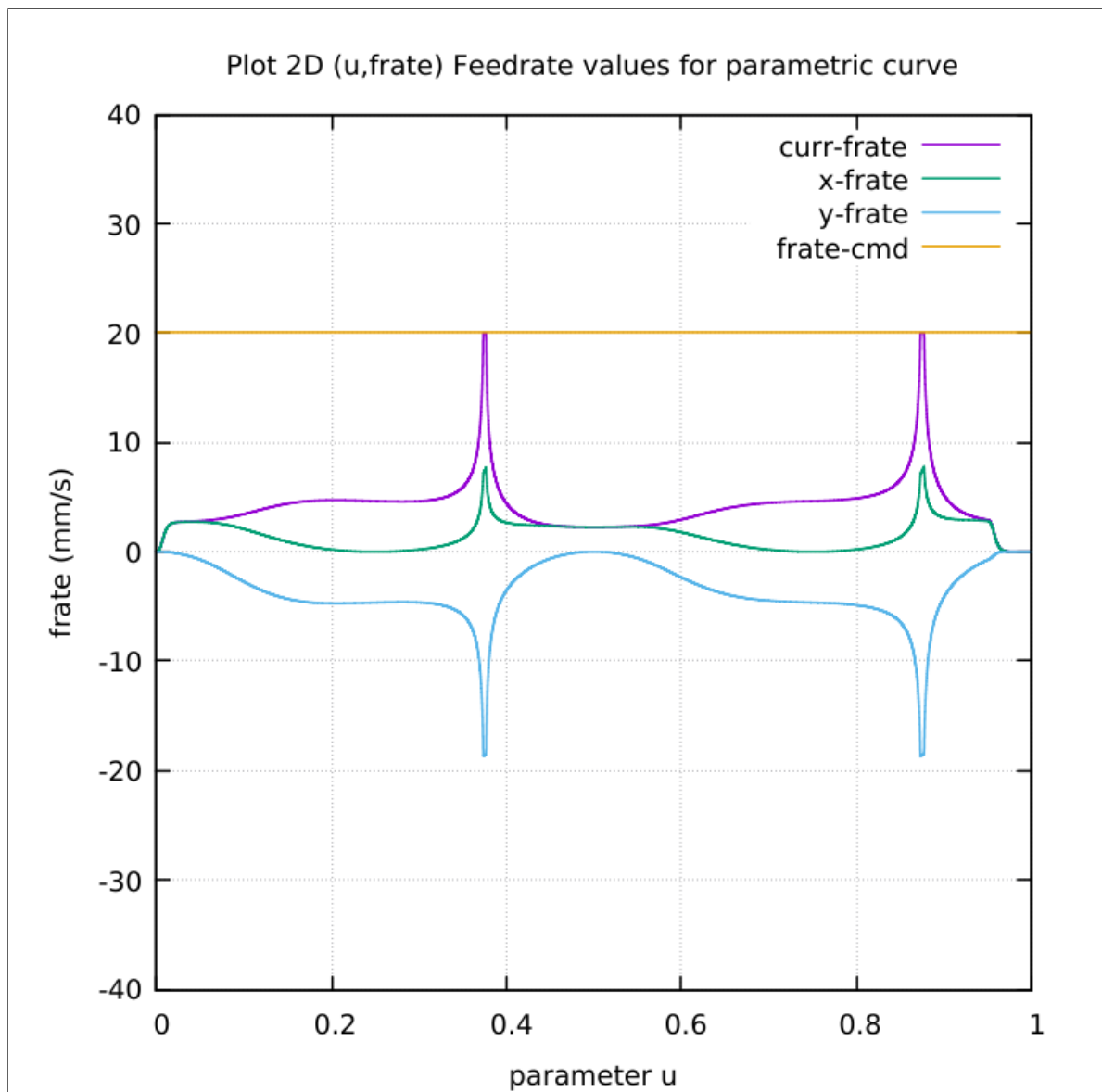


Table 5.42: Ribbon-10L FC20 u versus x-y-curr feedrate profile

### 5.4.18 Ribbon-10L FC20 x-y and colored feedrate profile

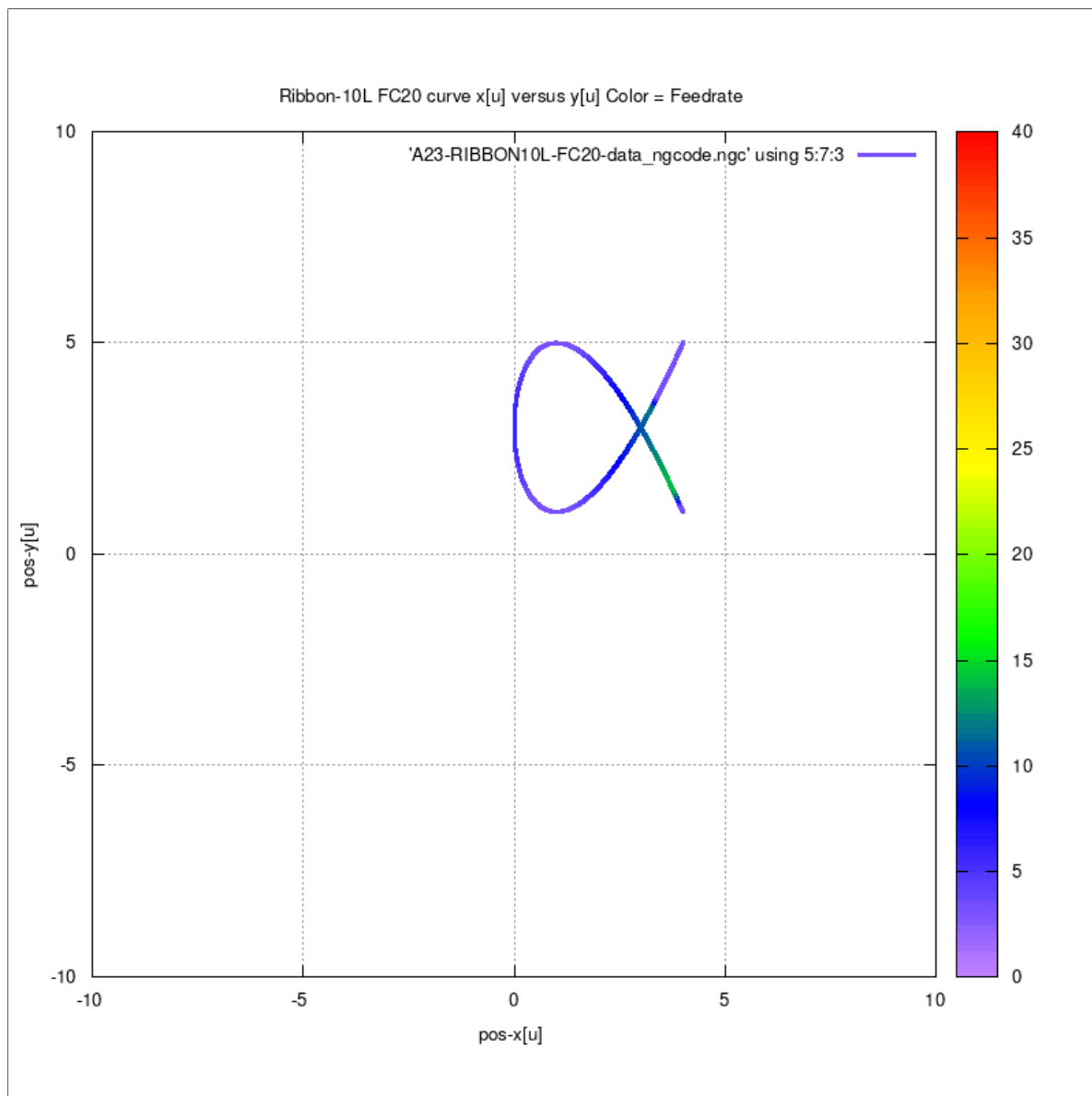


Table 5.43: Ribbon-10L FC20 x-y and colored feedrate profile

### 5.4.19 Ribbon-100L FC20 u versus x-y-curr feedrate profile

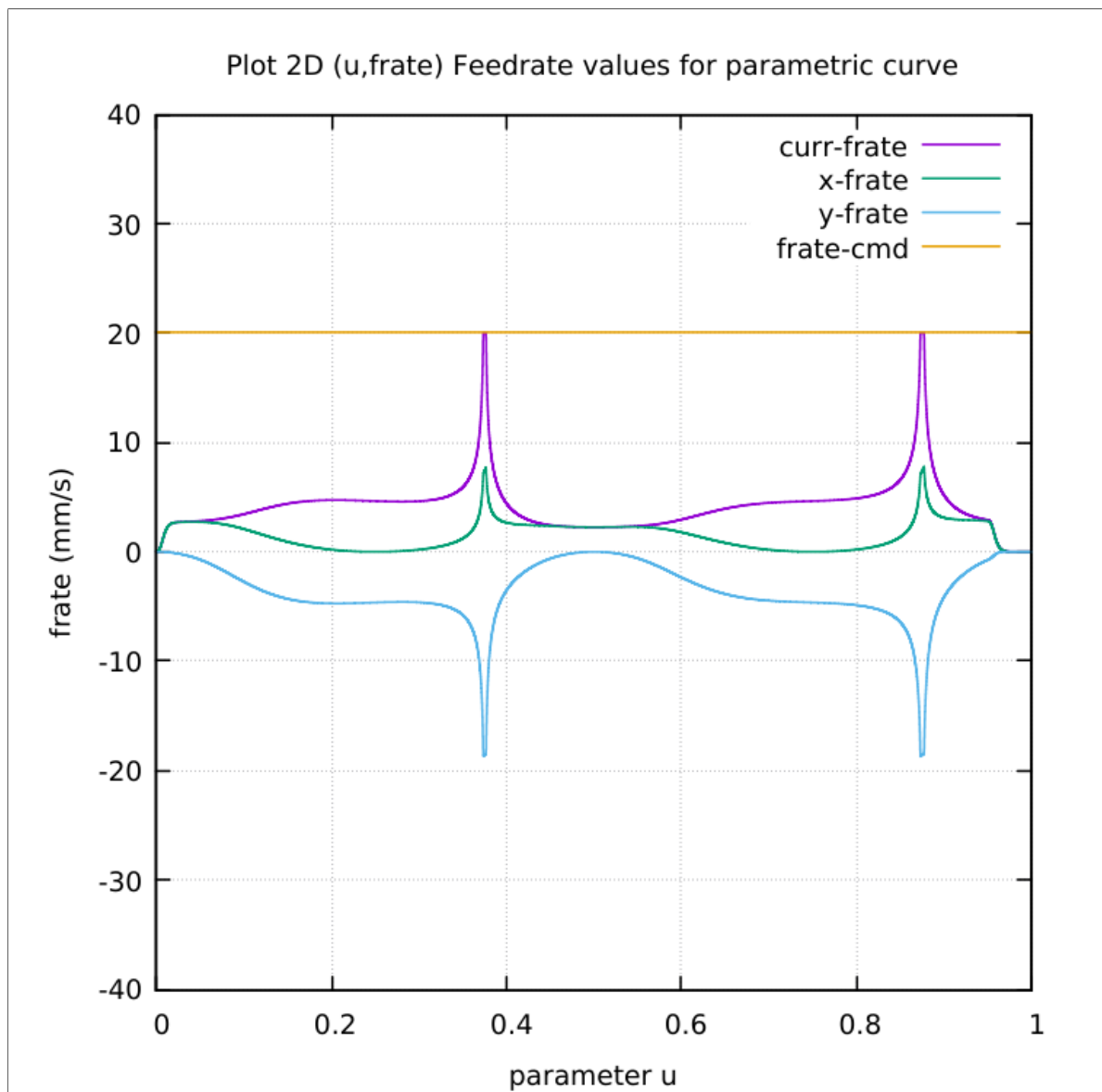


Table 5.44: Ribbon-100L FC20 u versus x-y-curr feedrate profile



### 5.4.20 Ribbon-100L FC20 x-y and colored feedrate profile

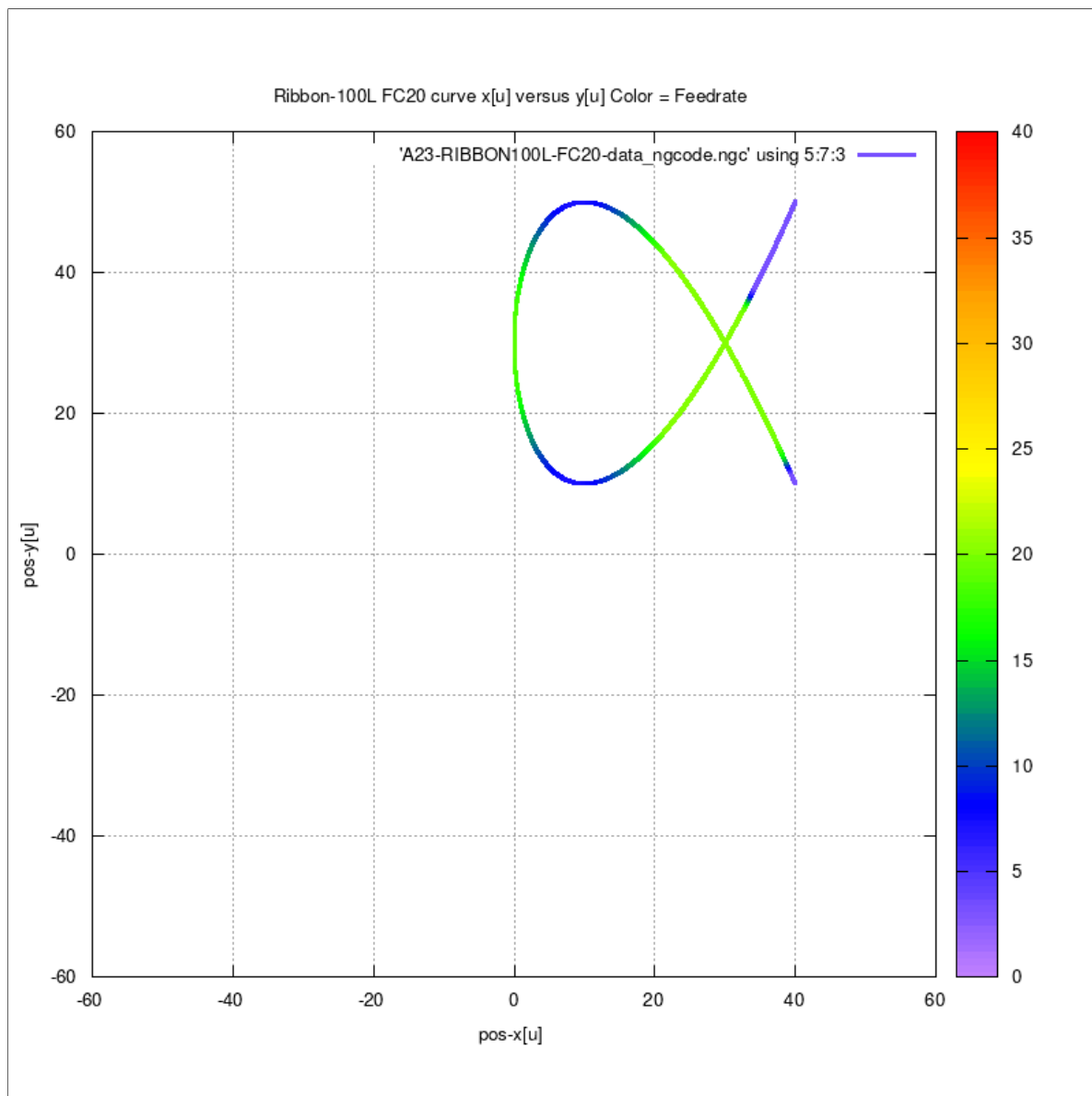


Table 5.45: Ribbon-100L FC20 x-y and colored feedrate profile

## 5.5 Error per unit length traversed

### 5.5.1 FC10 - Error per unit length traversed

Date: 2023-06-16 Author: wruslandr@gmail.com								
Total Interpolated Points for parametric curves								
CURVE	Total Interpolated Points					FC10	FC10	FC10
	FC10	FC20	FC25	FC30	FC40	Total curve length	Total error	error/length
Teardrop	10261	7599	7385	7347	7347	101.835673217	0.005809000	5.704288E-05
Butterfly	35656	18029	14577	12343	9732	356.074702570	0.001938860	5.445093E-06
Ellipse	21575	7599	9448	8338	7351	215.649935852	0.002990952	1.386948E-05
Skewed-Astroid	116194	58102	46483	38738	29056	445.714285882	0.000516368	1.158519E-06
Circle	49641	24822	19859	16549	12413	496.378581315	0.001093914	2.203790E-06
AstEpi	76275	38169	30563	25499	19184	426.262247842	0.000840373	1.971493E-06
Snailshell	15621	9883	8935	8370	7766	138.559540611	0.005115139	3.691654E-05
SnHyp	38672	20223	16618	11497	8889	478.987086578	0.002846873	5.943528E-06
Ribbon-10L	7351	7352	7352	7353	7353	15.210795863	0.007331687	4.820055E-04
Ribbon-100L	7480	7348	7349	7349	7350	152.097354848	0.007227414	4.751834E-05

Table 5.46: FC10 - Error per unit length for all parametric curves

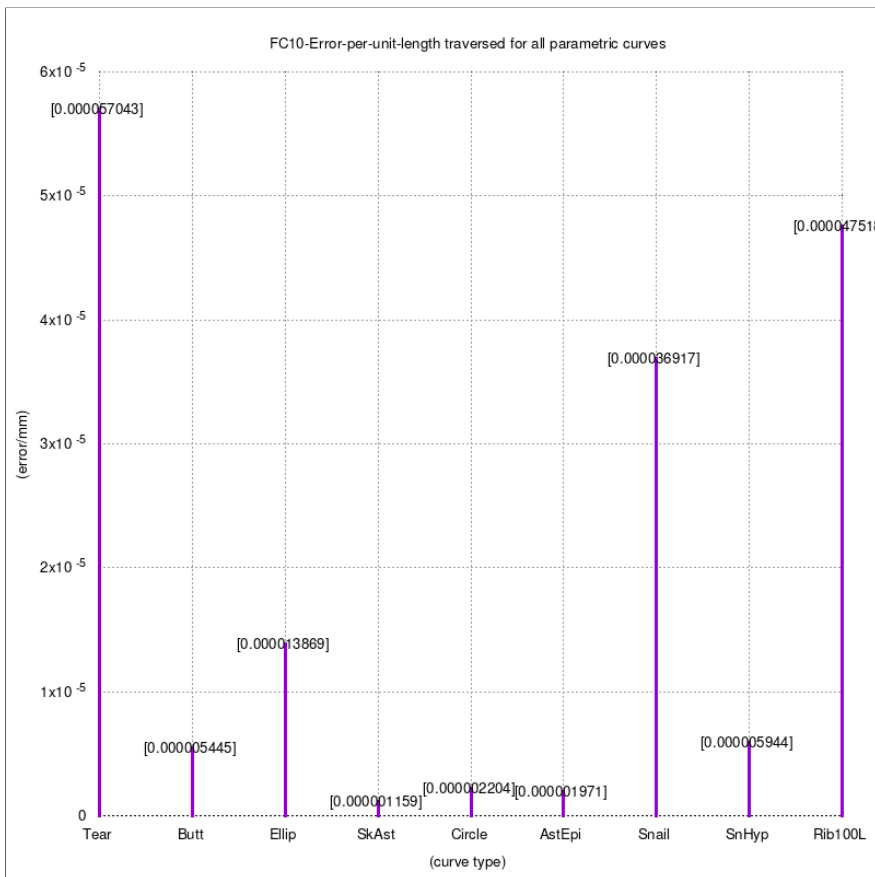


Table 5.47: FC10 - Error per unit length for all parametric curves

### 5.5.2 FC20 - Error per unit length traversed

Date: 2023-06-16		Total Interpolated Points for parametric curves						
Author: <a href="mailto:wruslandr@gmail.com">wruslandr@gmail.com</a>								
CURVE	Total Interpolated Points					FC20	FC20	FC20
	FC10	FC20	FC25	FC30	FC40	Total curve length	Total error	error/length
Teardrop	10261	7599	7385	7347	7347	101.841865570	0.007140807	7.011662E-05
Butterfly	35656	18029	14577	12343	9732	356.073710900	0.003534046	9.925040E-06
Ellipse	21575	7599	9448	8338	7351	215.649935852	0.007140807	3.311296E-05
Skewed-Astroid	116194	58102	46483	38738	29056	445.714285537	0.001032591	2.316711E-06
Circle	49641	24822	19859	16549	12413	496.377158168	0.001093914	2.203796E-06
AstEpi	76275	38169	30563	25499	19184	426.262239690	0.001641843	3.851720E-06
Snailshell	15621	9883	8935	8370	7766	138.561390573	0.006269762	4.524898E-05
SnHyp	38672	20223	16618	11497	8889	478.998699413	0.004002975	8.356964E-06
Ribbon-10L	7351	7352	7352	7353	7353	15.210689035	0.007330650	4.819407E-04
Ribbon-100L	7480	7348	7349	7349	7350	152.102890678	0.007334489	4.822058E-05

Table 5.48: FC20 - Error per unit length for all parametric curves

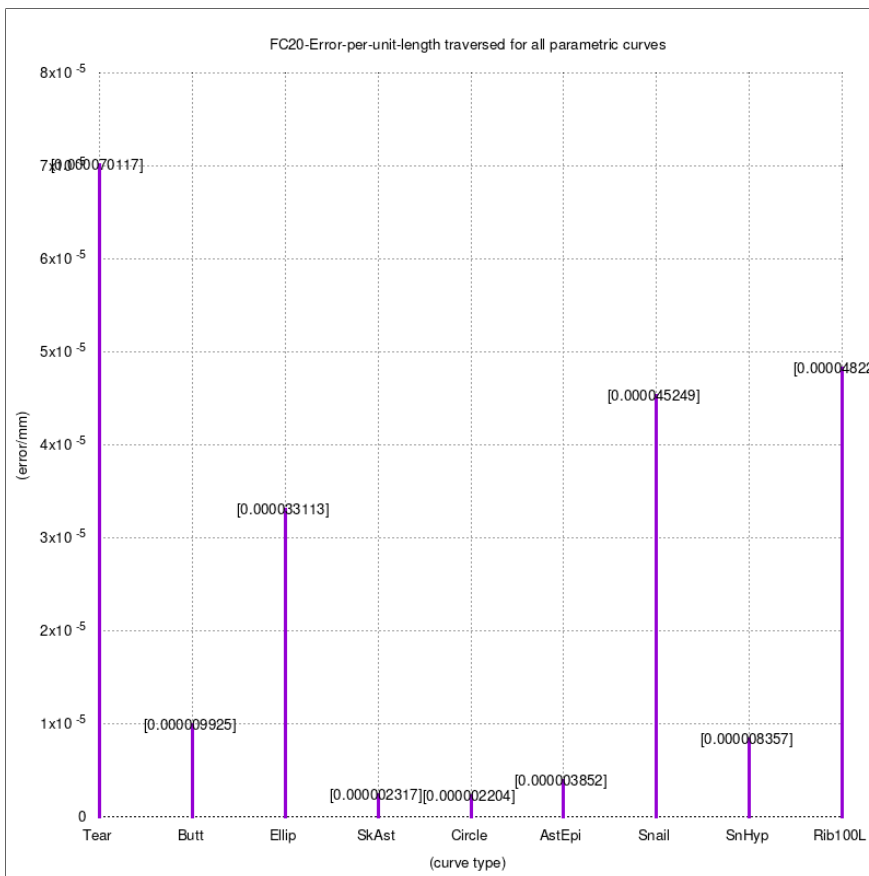


Table 5.49: FC20 - Error per unit length for all parametric curves

### 5.5.3 FC25 - Error per unit length traversed

Date: 2023-06-16 Author: <a href="mailto:wruslandr@gmail.com">wruslandr@gmail.com</a>		Total Interpolated Points for parametric curves						
CURVE	Total Interpolated Points					FC25	FC25	FC25
	FC10	FC20	FC25	FC30	FC40	Total curve length	Total error	error/length
Teardrop	10261	7599	7385	7347	7347	101.834772771	0.007301198	7.169651E-05
Butterfly	35656	18029	14577	12343	9732	356.072310198	0.004230857	1.188202E-05
Ellipse	21575	7599	9448	8338	7351	215.644021538	0.005927619	2.748798E-05
Skewed-Astroid	116194	58102	46483	38738	29056	445.714282439	0.001290613	2.895606E-06
Circle	49641	24822	19859	16549	12413	496.396441759	0.002187640	4.407042E-06
AstEpi	76275	38169	30563	25499	19184	426.262236290	0.002020019	4.738911E-06
Snailshell	15621	9883	8935	8370	7766	138.560655481	0.006558063	4.732991E-05
SnHyp	38672	20223	16618	11497	8889	479.006371543	0.004459255	9.309386E-06
Ribbon-10L	7351	7352	7352	7353	7353	15.209447637	0.007330192	4.819499E-04
Ribbon-100L	7480	7348	7349	7349	7350	152.132129168	0.007334570	4.821184E-05

Table 5.50: FC25 - Error per unit length for all parametric curves

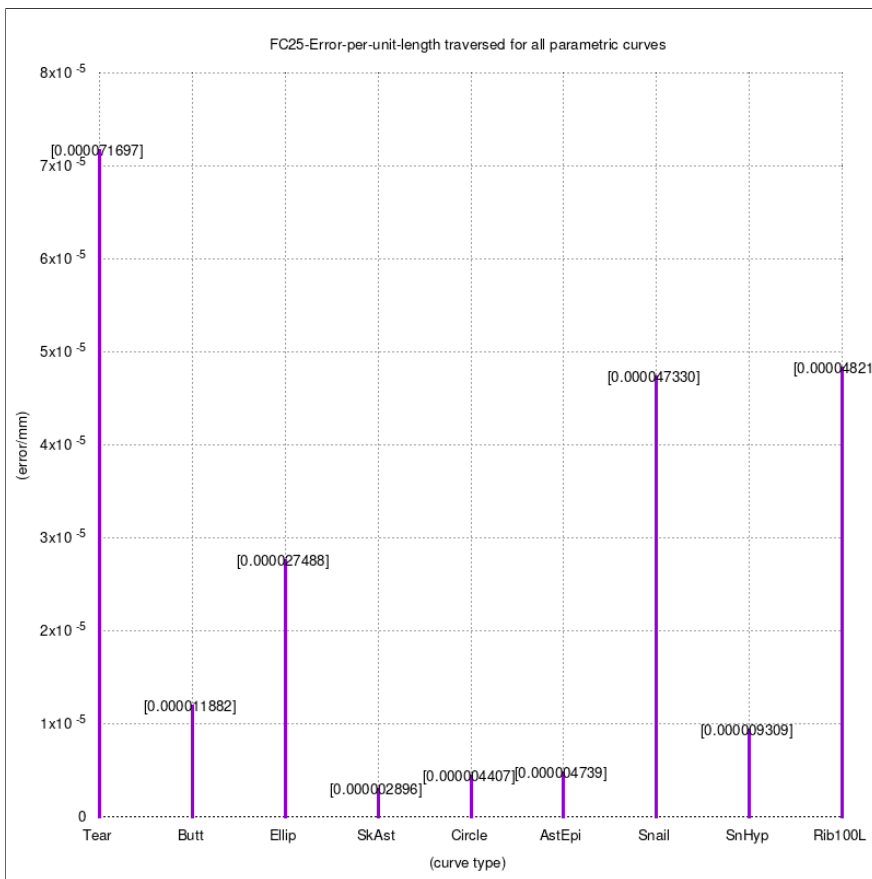


Table 5.51: FC25 - Error per unit length for all parametric curves

### 5.5.4 FC30 - Error per unit length traversed

Date: 2023-06-16		Total Interpolated Points for parametric curves						
Author: wruslandr@gmail.com								
CURVE	Total Interpolated Points					FC30	FC30	FC30
	FC10	FC20	FC25	FC30	FC40	Total curve length	Total error	error/length
Teardrop	10261	7599	7385	7347	7347	101.859566601	0.007336794	7.202852E-05
Butterfly	35656	18029	14577	12343	9732	356.072793009	0.004846583	1.361121E-05
Ellipse	21575	7599	9448	8338	7351	215.647842617	0.006561982	3.042916E-05
Skewed-Astroid	116194	58102	46483	38738	29056	445.714284621	0.001548669	3.474578E-06
Circle	49641	24822	19859	16549	12413	496.375730563	0.002734540	5.509012E-06
AstEpi	76275	38169	30563	25499	19184	426.262229552	0.002390002	5.606882E-06
Snailshell	15621	9883	8935	8370	7766	138.560164183	0.006764497	4.881993E-05
SnaHyp	38672	20223	16618	11497	8889	0.000000000	0.000000000	0.000000E+00
Ribbon-10L	7351	7352	7352	7353	7353	15.213913854	0.007330844	4.818513E-04
Ribbon-100L	7480	7348	7349	7349	7350	152.110308628	0.007333765	4.821346E-05

Table 5.52: FC30 - Error per unit length for all parametric curves

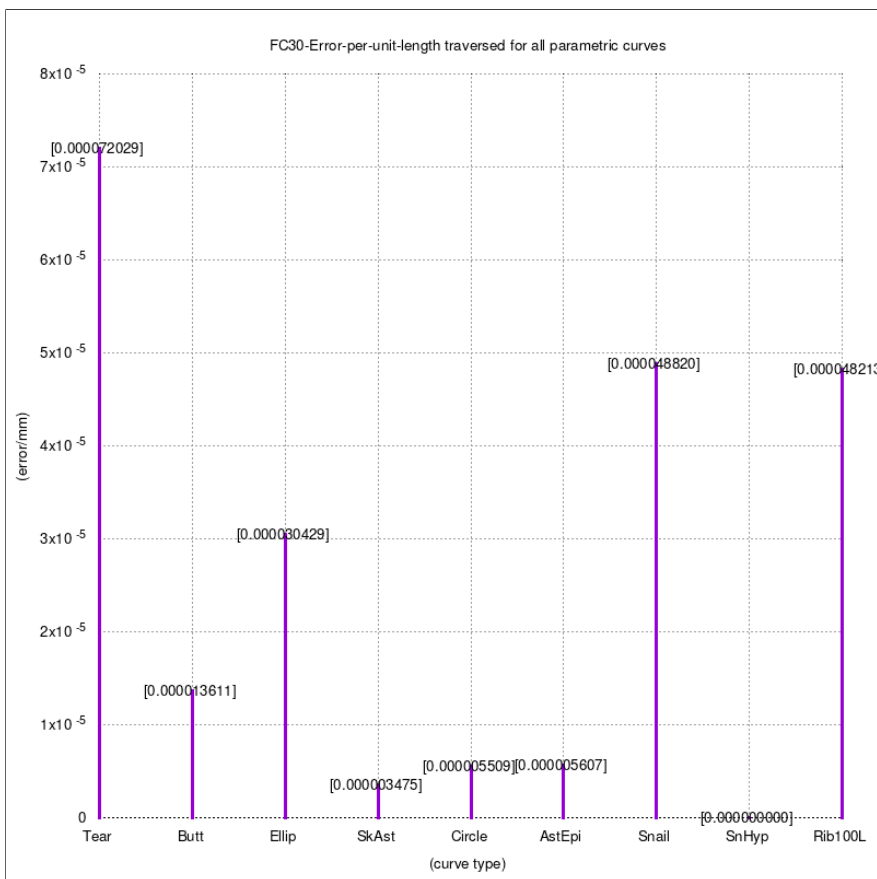


Table 5.53: FC30 - Error per unit length for all parametric curves

### 5.5.5 FC40 - Error per unit length traversed

Date: 2023-06-16		Total Interpolated Points for parametric curves						
Author: wruslandr@gmail.com								
CURVE	Total Interpolated Points					FC40	FC40	FC40
	FC10	FC20	FC25	FC30	FC40	Total curve length	Total error	error/length
Teardrop	10261	7599	7385	7347	7347	101.835560839	0.007335147	7.202933E-05
Butterfly	35656	18029	14577	12343	9732	356.073152673	0.005851473	1.643335E-05
Ellipse	21575	7599	9448	8338	7351	215.643855145	0.007331841	3.399977E-05
Skewed-Astroid	116194	58102	46483	38738	29056	445.714283175	0.002060000	4.621795E-06
Circle	49641	24822	19859	16549	12413	496.394293434	0.004374701	8.812956E-06
AstEpi	76275	38169	30563	25499	19184	426.262233355	0.003111370	7.299192E-06
Snailshell	15621	9883	8935	8370	7766	138.559886203	0.007045829	5.085042E-05
SnaHyp	38672	20223	16618	11497	8889	0.000000000	0.000000000	0.000000E+00
Ribbon-10L	7351	7352	7352	7353	7353	15.211915247	0.007330108	4.818662E-04
Ribbon-100L	7480	7348	7349	7349	7350	152.139353248	0.007333840	4.820475E-05

Table 5.54: FC40 - Error per unit length for all parametric curves

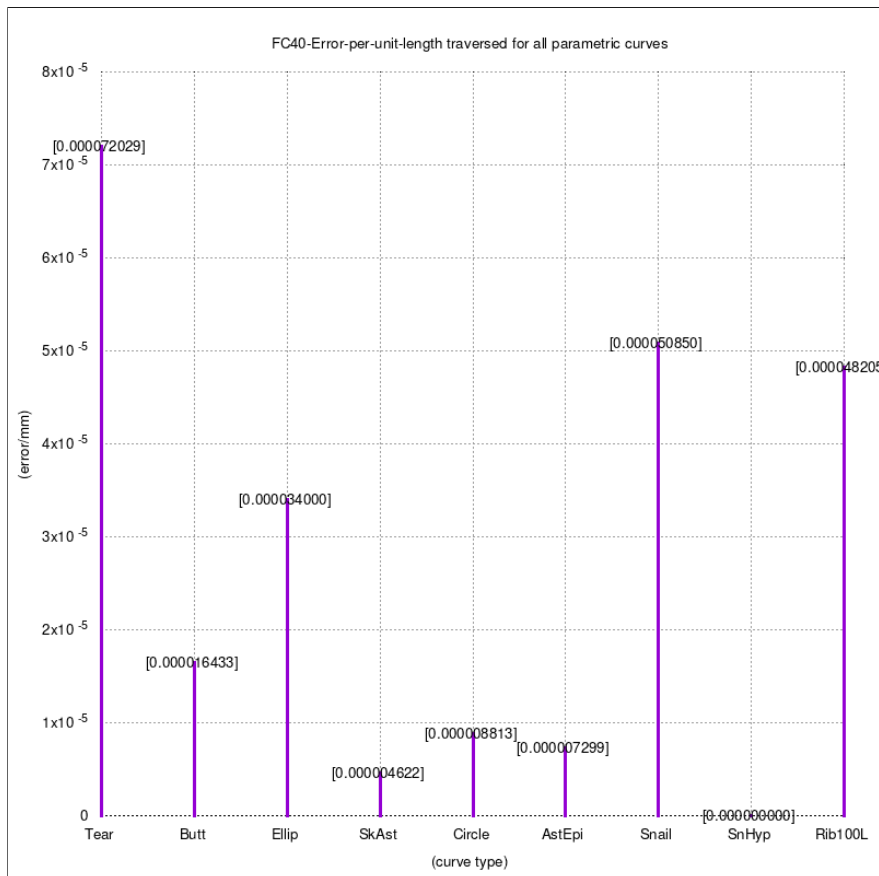


Table 5.55: FC40 - Error per unit length for all parametric curves

## 5.6 Histogram of Interpolated Points



### 5.6.1 Teardrop histogram of interpolated points

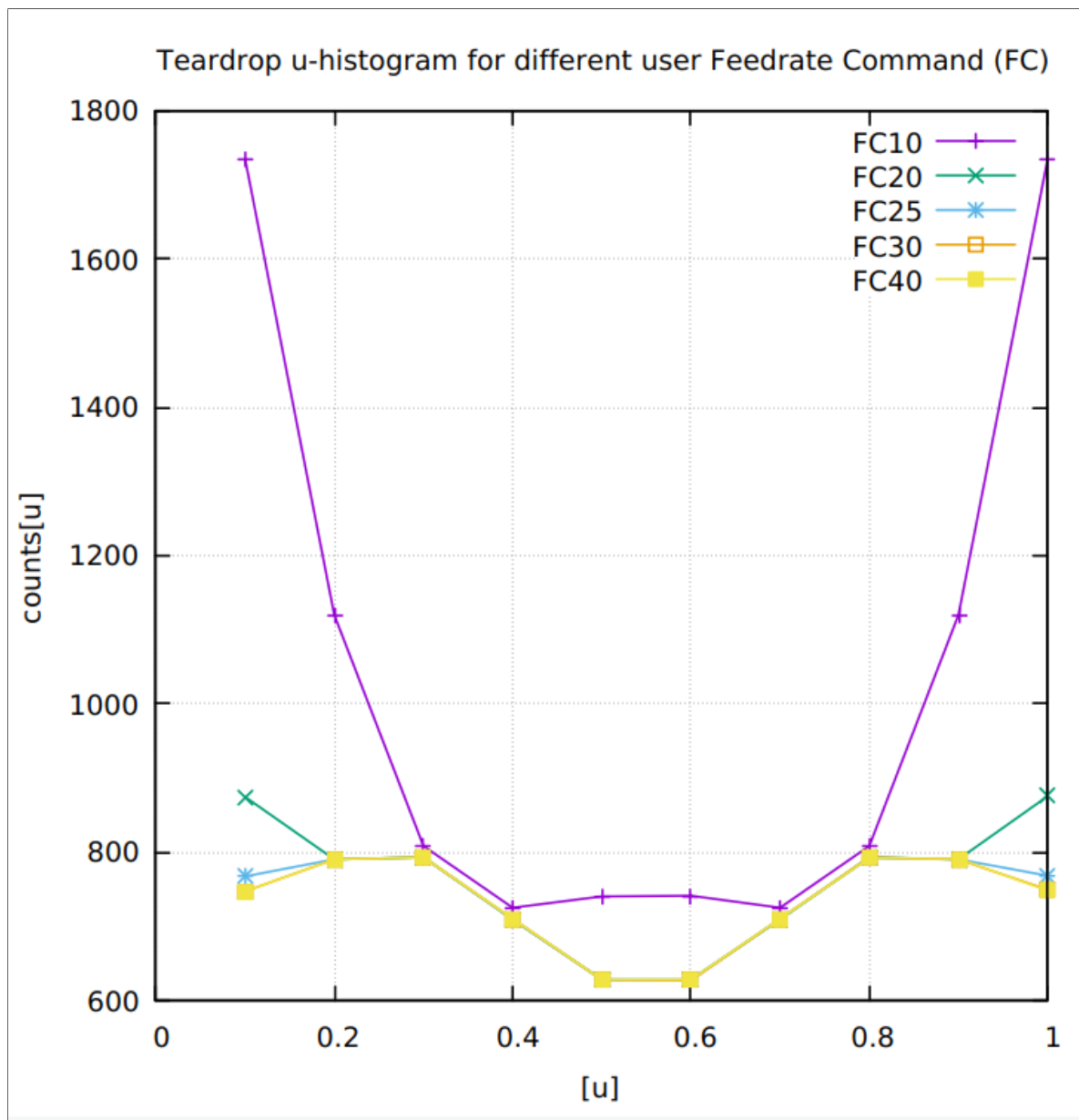


Table 5.56: Teardrop histogram of interpolated points

### 5.6.2 Butterfly histogram of interpolated points

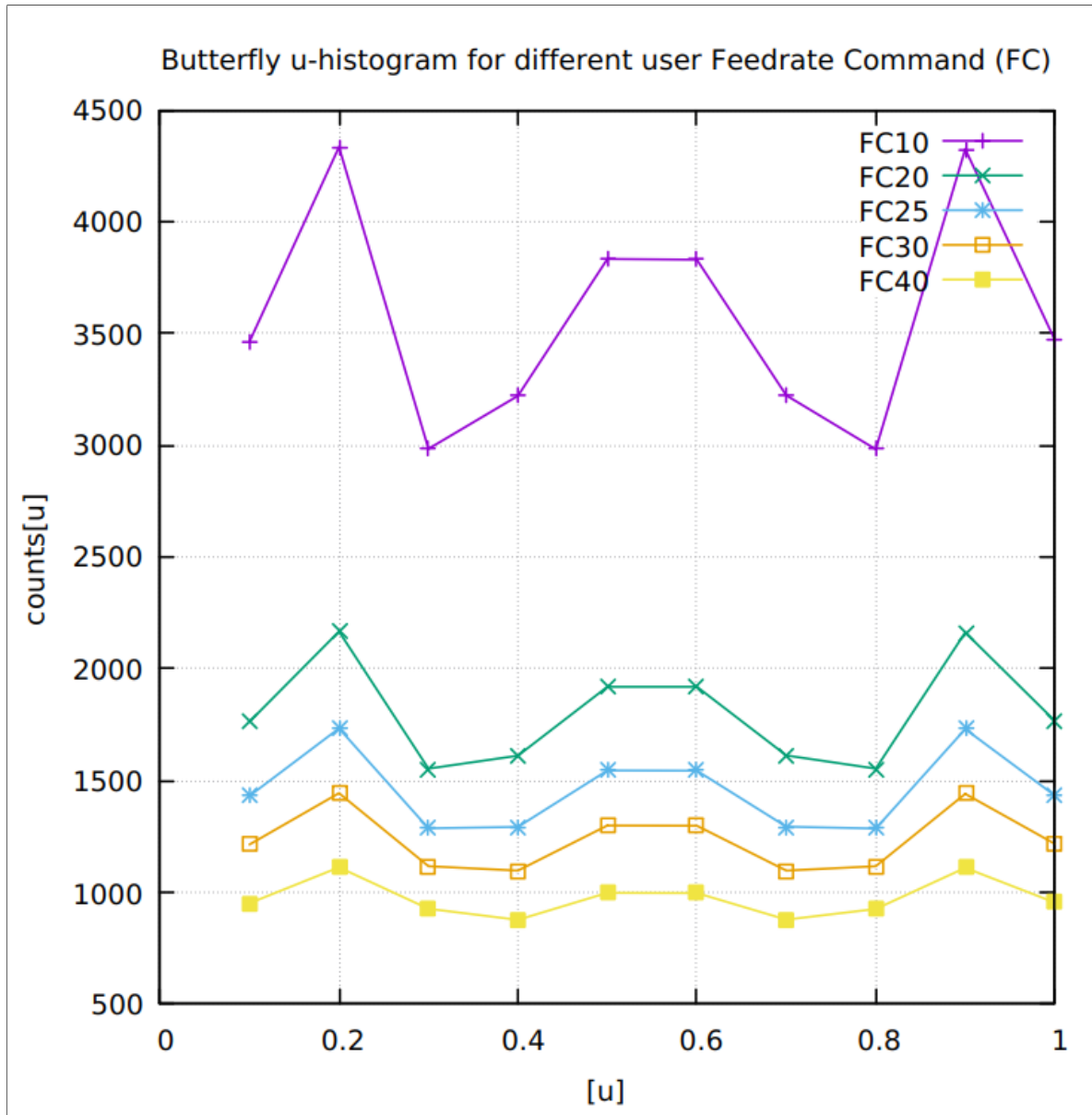


Table 5.57: Butterfly histogram of interpolated points

### 5.6.3 Ellipse histogram of interpolated points

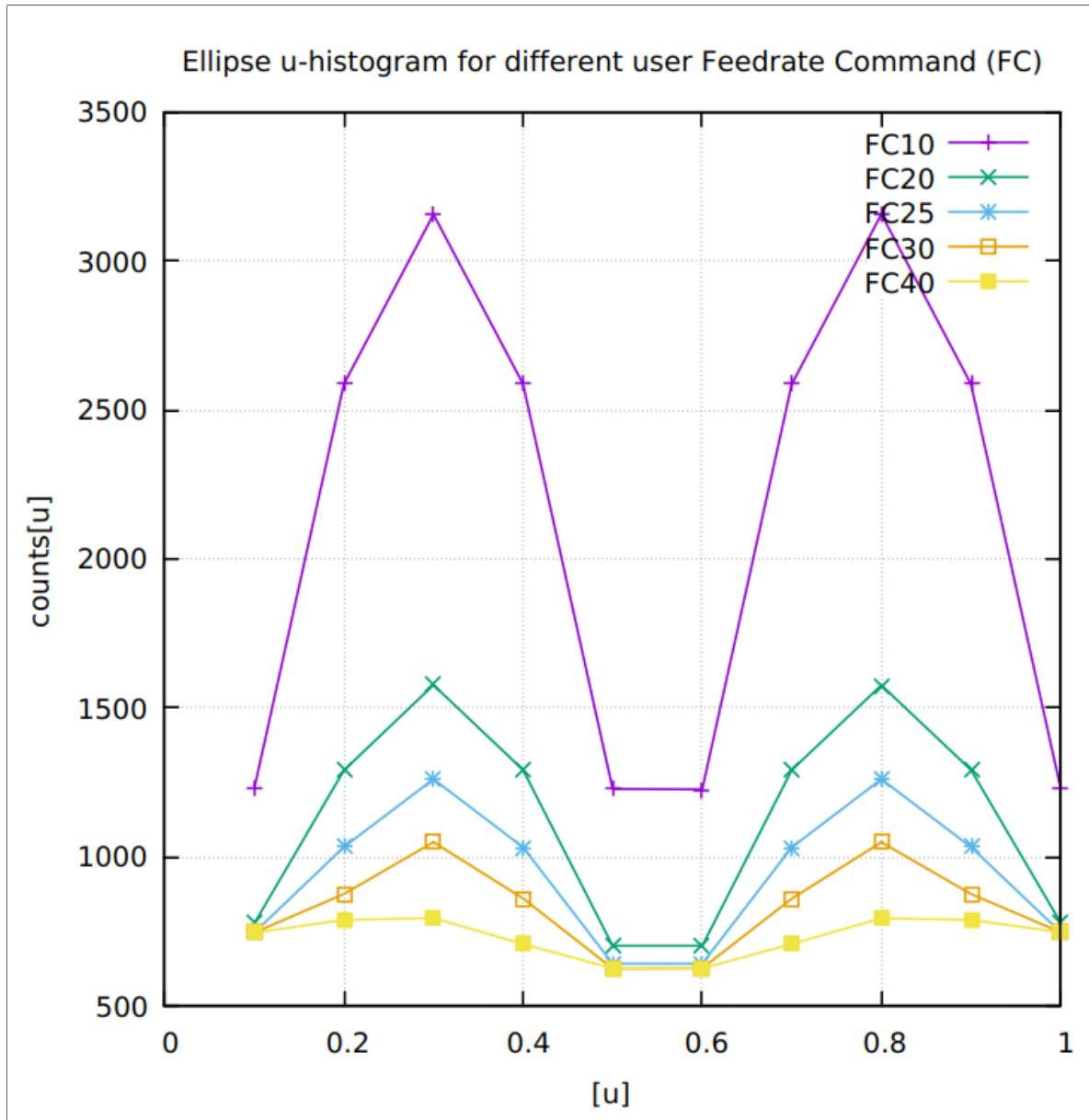


Table 5.58: Ellipse histogram of interpolated points

### 5.6.4 Skewed-Astroid histogram of interpolated points

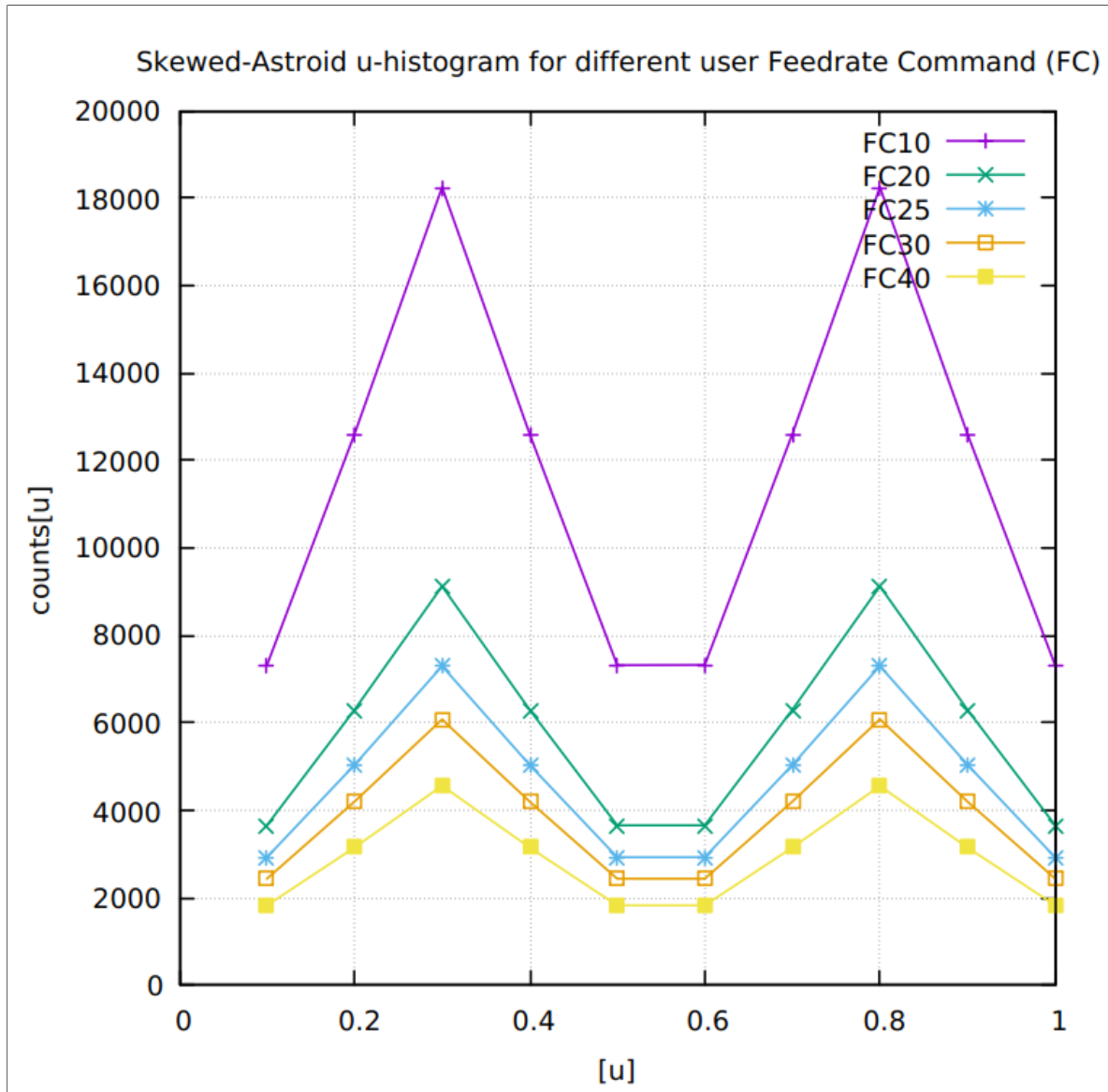


Table 5.59: Skewed-Astroid histogram of interpolated points

### 5.6.5 Circle histogram of interpolated points

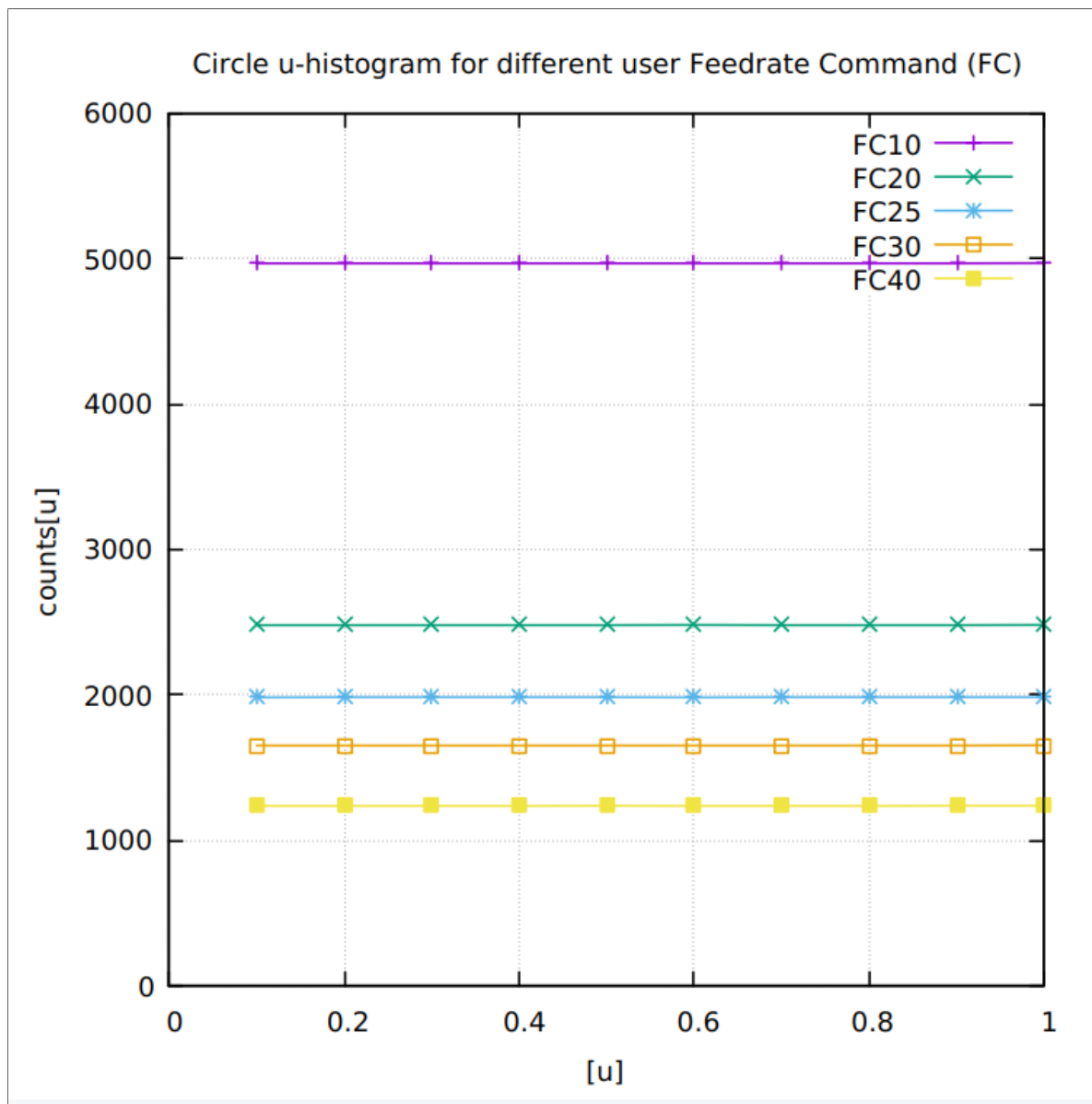


Table 5.60: Circle histogram of interpolated points

### 5.6.6 AstEpi histogram of interpolated points

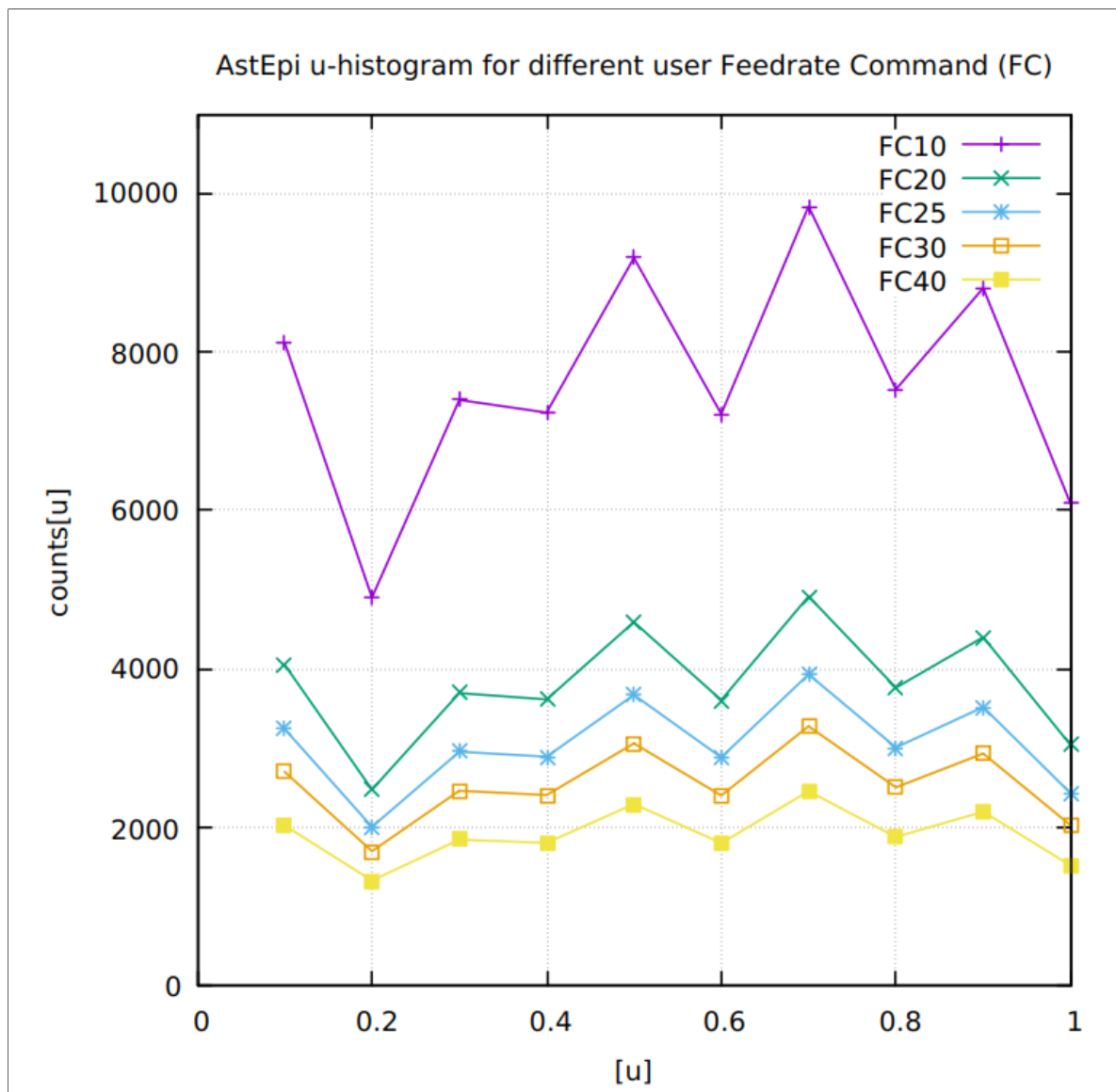


Table 5.61: AstEpi histogram of interpolated points

### 5.6.7 Snailshell histogram of interpolated points

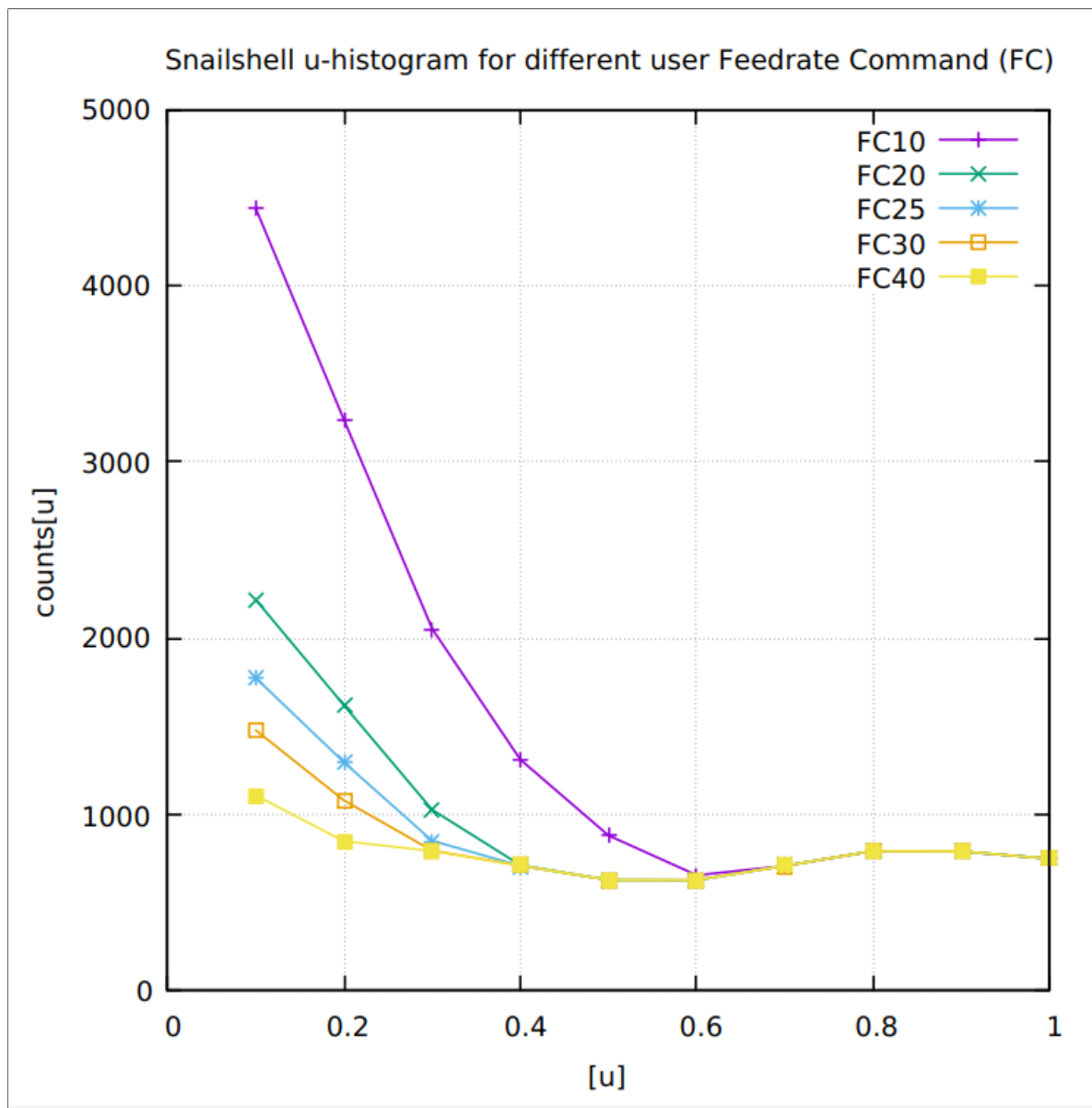


Table 5.62: Snailshell histogram of interpolated points

### 5.6.8 SnaHyp histogram of interpolated points

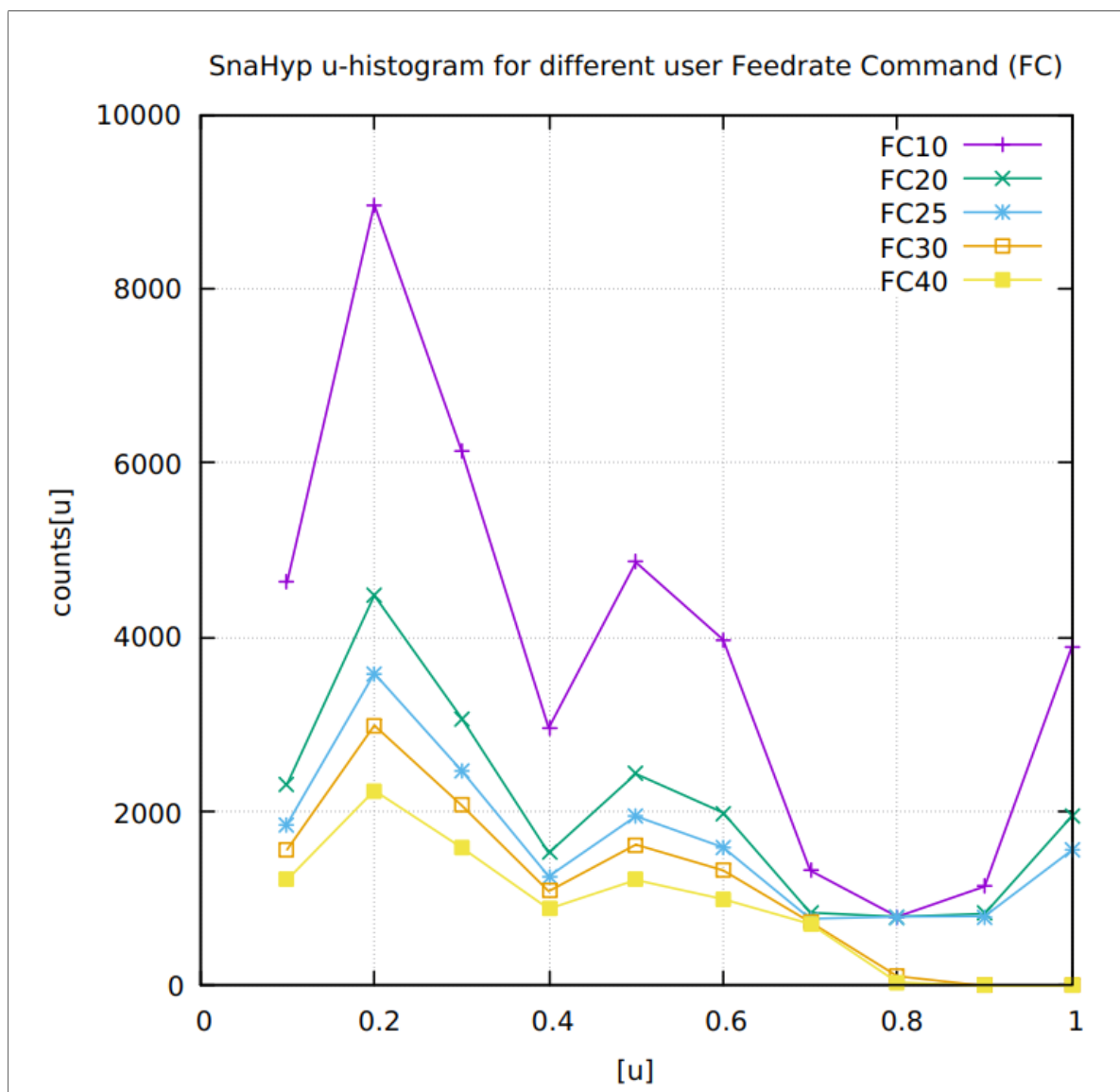


Table 5.63: SnaHyp histogram of interpolated points



### 5.6.9 Ribbon-10L histogram of interpolated points

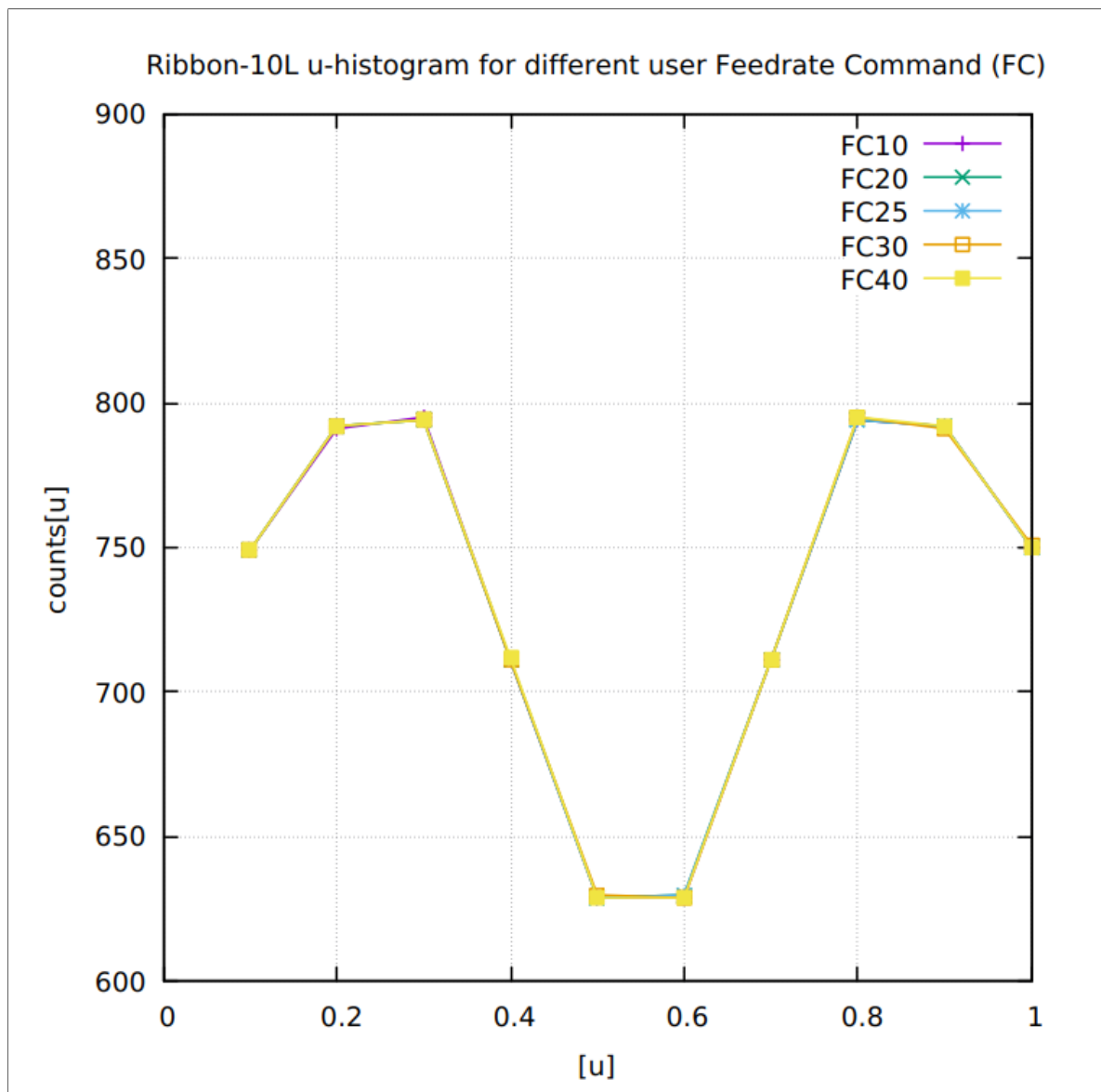


Table 5.64: Ribbon-10L histogram of interpolated points

### 5.6.10 Ribbon-100L histogram of interpolated points

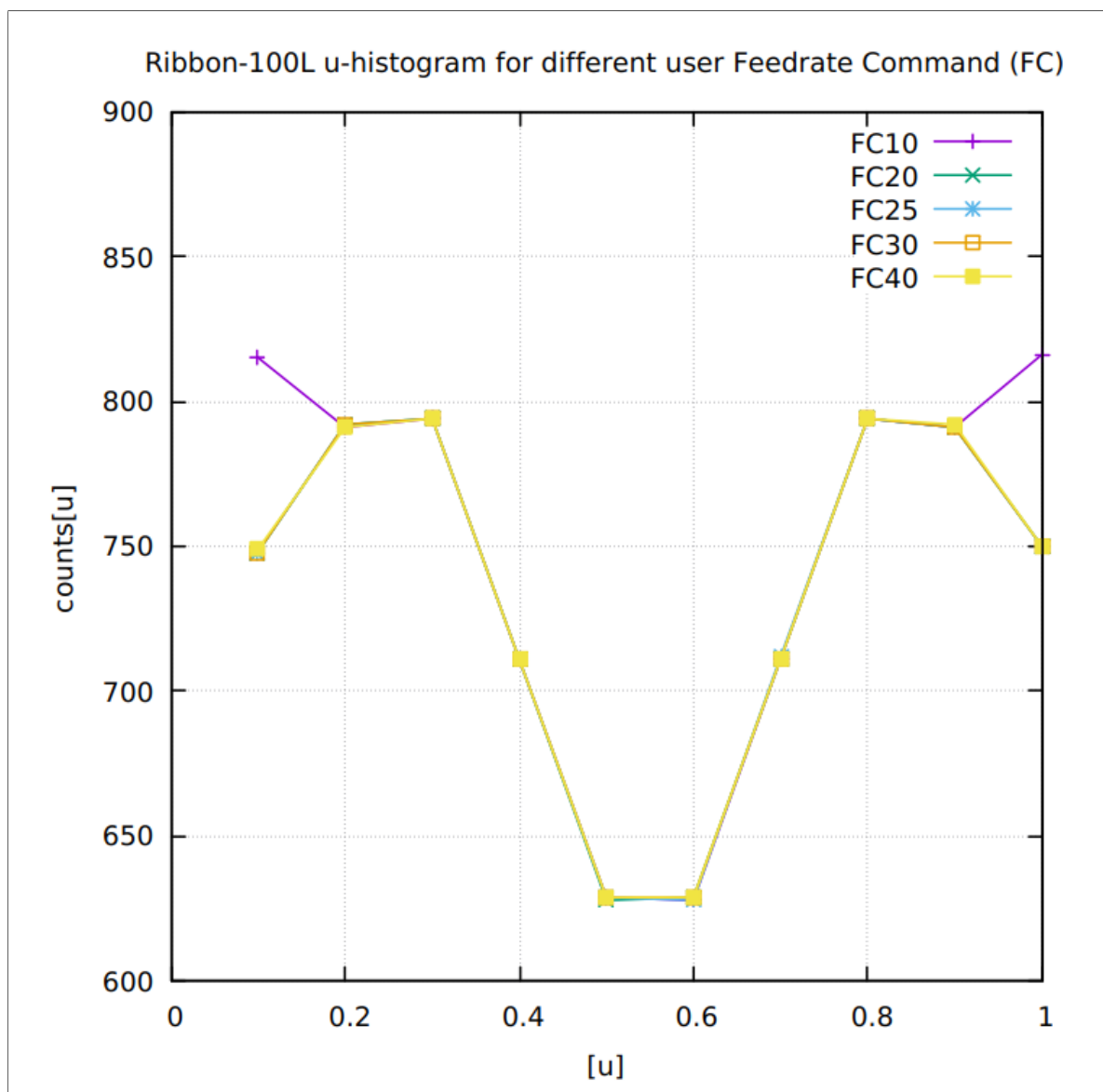


Table 5.65: Ribbon-100L histogram of interpolated points

## 5.7 Experimental Validation

Run G-CODES

# 6 Analyses and Discussion

# 7 Conclusion

# 1 Appendices