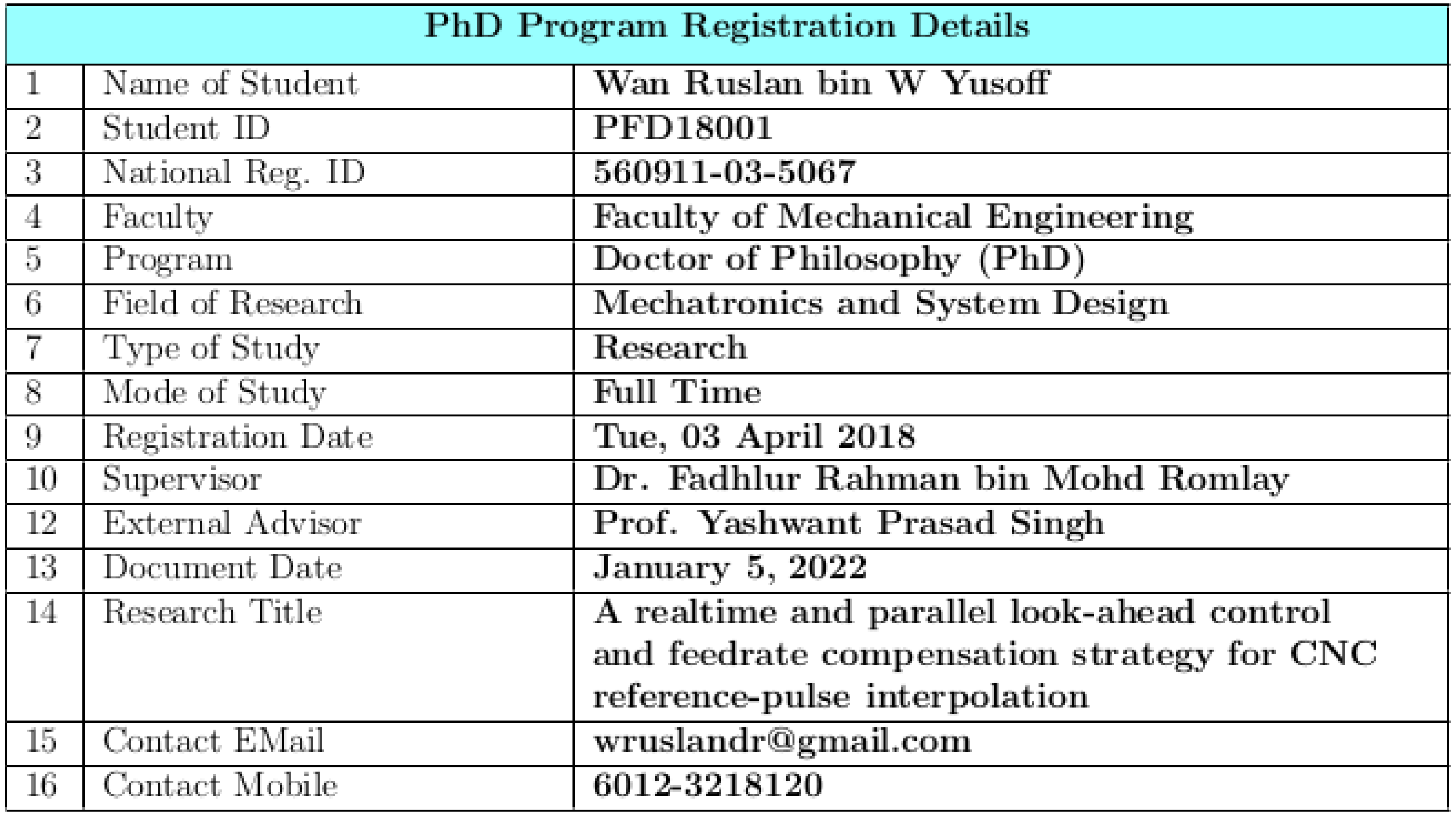
PHD PROGRESS REPORT – UMP SEMESTER 1 SESSION (2022/2023) **Date: 26 Dec 2022**

Faculty of Mechanical and Automotive 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	**January 5, 2022**
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**

THE PROBLEM STATEMENT - PARAMETRIC CURVE INTERPOLATION

The problem of parametric curve interpolation can be expressed as,

> **given the following five(5) current values:**
>
> (1) the parametric curve C (u) ;
> (2) the command feedrate F;
> (3) the interpolation period T;
> (4) the current (k-th interpolation period) motion status: that is, feedrate (frate[k]) and acceleration (accn[k]) ;
> (5) the current reference point C(u[k]) .
>
> **then calculate the reference point C(u[k+1]) for the next interpolation period. This cycle repeats until completion.**

And the **result** should be subject to machine dynamics constraints and chord error tolerance (epsilon[k]) : **The constraints comprise:**

> (6) Axial velocities and accelerations should be limited to avoid saturating the drive;
>
> (7) The jerk should be limited to avoid the excitation of vibrations in components in the machine assembly;
>
> (8) The chord error increases with feedrate and curvature, so the feedrate should be limited to achieve high interpolation accuracy, while the productivity should be kept as high as possible.
>
> (9) The interpolator plans S-shaped feedrate profiles at the beginning and end of the parametric curve.

STUDY CONTRIBUTIONS

> (1) Introduce a constraint on the chord error (epsilon[k]). This significantly decreases the number of interpolated points. The less number of interpolated points means increase productivity, since there are less total points to process..
>
> (2) Imposing a maximum chord error constraint increases chord error. However, the trade-off is an increase in smoothness (curvature) of feedrate (frate[k]) profiles and thus, results in reduction of machine jerk and acceleration.

CONSTRAINTS ON FEEDRATE LIMITS

At each interpolation point [k], the **effective feedrate limit is the minimum** among the following four(4) feedrate constraints:

Feedrate constraint 1 (FRLimit-1)

The feedrate should not exceed the command feedrate, as the command feedrate is specified by the user with the consideration of the process, machine dynamics and capabilities.

FRLimit-1 = Command_feedrate.

Feedrate constraint 2 (FRLimit-2)

The second constraint depends on maximum axial velocity.

FRLimit-2 = Minimum of [Vx/|alpha|, Vy/|beta|] where

Vx = x-axis velocity and alpha = ratio magnitude (x-velocity/curve velocity)
Vy = y-axis velocity and beta = ratio magnitude (y-velocity/curve velocity)

Feedrate constraint 3 (FRLimit-3)

The third constraint depends on contour accuracy, meaning on chord error (eps), curvature (rho) and interpolation time (T).

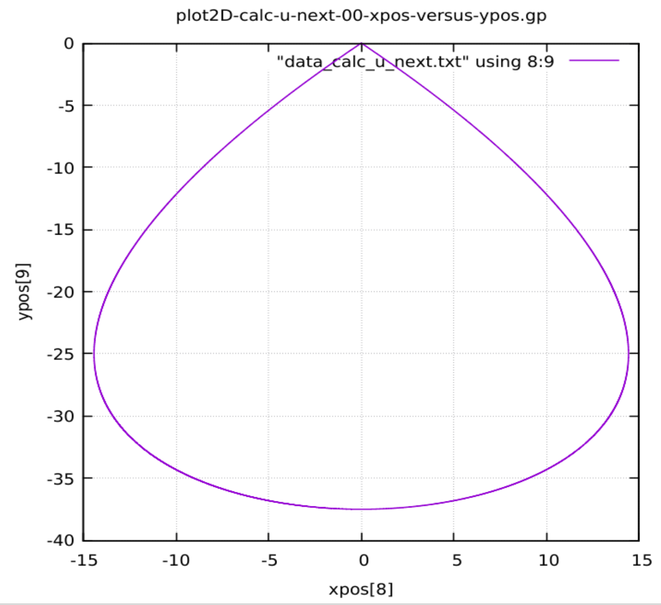
FRLimit-3 = (2/T)square-root(2.rho.eps – eps.eps)

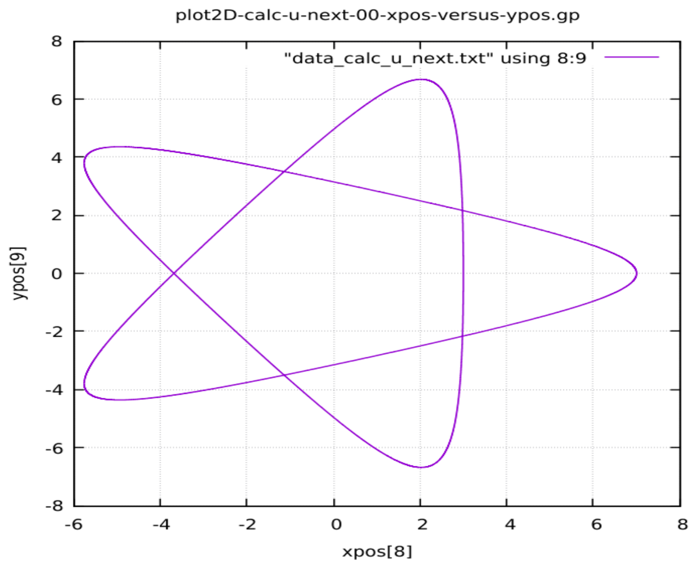
Feedrate constraint 4 (FRLimit-4)

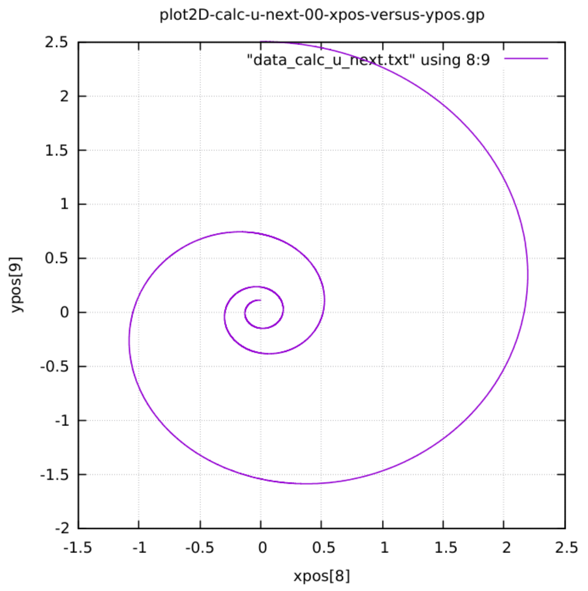
The fourth feedrate limitation depends on normal acceleration(An) and feedrate (F),

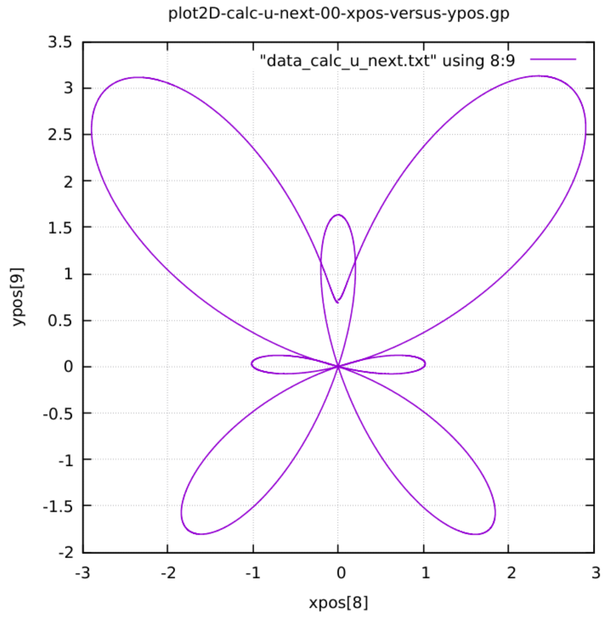
FRLimit-4 = Minimum of [A, B] where

A = square-root[lamda.rho.(Ax) / (|beta|)]
B = square-root[lamda.rho.(Ay)/ / (|alpha|)]
(Ax) = x-axis acceleration
(Ay) = y-axis acceleration
lambda = safety factor in the range (0 to 1).


plot2D-calc-u-next-00-xpos-versus-ypos.gp
"data_calc_u_next.txt" using 8:9
ypos[9]
xpos[8]
0
-5
-10
-15
-20
-25
-30
-35
-40
-15
-10
-5
0
5
10
15


plot2D-calc-u-next-00-xpos-versus-ypos.gp
"data_calc_u_next.txt" using 8:9
ypos[9]
xpos[8]
8
6
4
2
0
-2
-4
-6
-8
-6
-4
-2
0
2
4
6
8


plot2D-calc-u-next-00-xpos-versus-ypos.gp
"data_calc_u_next.txt" using 8:9
ypos[9]
xpos[8]
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5


plot2D-calc-u-next-00-xpos-versus-ypos.gp
"data_calc_u_next.txt" using 8:9
ypos[9]
xpos[8]
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-3
-2
-1
0
1
2
3

INTERPOLATOR ALGORITHMS

The algorithms in this study are **cumulative** starting from Algorithm 1 until Algorithm 4. The algorithms were executed sequentially for each parametric curve. The curves comprise Teardrop (top left), Hypotropoid (top right), Snailshell (bottom left) and Butterfly (bottom right). The functions executed in each algorithm are described in the table below.

Algorithm 1	**Algorithm 2**	**Algorithm 3**	**Algorithm 4**
Initialize variables in algorithm.	Additions to Algorithm 1.	Additions to Algorithm 2.	Additions to Algorithm 3.
Apply the second order Taylor's interpolation approximation to the parametric curve.	CASE A : if (curr_frate_limit > curr_frate), then increase curr_frate, and increase curr_tang_accn.		
Apply rising S-curve from u = 0.0 to u = 0.10. Apply falling S-curve from u = 0.95 to u = 1.00.	CASE B : if (curr_frate_limit < curr_frate), then decrease curr_frate, and decrease curr_tang_accn.		
Fix the feedrate_command = 20.0; Calculate minimum and maximum tangential accelerations at (u);	CASE C: if (curr_frate_limit = curr_frate), then next_frate = curr_frate; and next_tang_accn = curr_tang_accn; Maintain values.		
Calculate the current feedrate limit at each u. Calculate the next feedrate based on current feedrate limit, current feedrate, maximum jerk and interpolation period at each u.	*Restricts feedrate to be strictly below feedrate limit at each u. *No restrictions or control on chord error (eps) for all u, meaning allows chord errors to be above and below error tolerance 10(-6).	*Decrease (push down) all chord errors (eps) to just below and very near to error tolerance 10(-6)	
Calculate the u_next using the previous u_next, next feedrate, and interpolation period. With the new u_next, calculate eps and rho.		*Calculate the next u (i.e. u_next) such that all chord errors (eps) are below error tolerance 10(-6).	*Increase (raise up) those chord errors very far below tolerance to near tolerance 10(-6), but not above tolerance.
Update new u = current u + u_next. **Repeat the loop until u = 1.00. Then exit the algorithm.**	**Repeat the loop, then exit the algorithm.**	**Repeat the loop, then exit the algorithm.**	*Generate NGC gcode file for the parametric curve. **Repeat the loop, then exit the algorithm.**

INTERPOLATION RUN RESULTS

The number of interpolated points for the different 2D parametric curves are provided in the table below.

Parametric curve type	Curve characteristics	Parametric Equations	Algo 1 Points	Algo 2 Points	Algo 3 Points	Algo 4 Points	Diff Algo 2 and Algo 3
Teardrop	Closed loop Single loop x-axis non symmetrical y-axis symmetrical	x = – 150.u + 450.u.u – 300.u.u.u y = – 150.u + 150.u.u; Parameter range u[0.0 :1.0]	5096	9204	7599	7344	1805 17 %
Hypotropoid	Closed loop Multiple loops x-axis symmetrical y-axis non symmetrical	x = 2.cos(u) + 5.cos(2.u/3) y = 2.sin(u) – 5.sin(2.u/3) Parameter range u[0.0 : 6.0 PI]	3443	9545	7347	7344	2198 23 %
Snailshell	Open loop Single curve scaled up x-axis non symmetrical y-axis non symmetrical	x = 10.[sin(2.u) / (u.u + 4)] y = 10.[cos(2.u) / (u.u + 4)] Parameter range u[0.0 : 3.0 PI]	706	14601	7351	7346	7250 50 %
Butterfly	Closed loop Multiple loops x-axis non symmetrical y-axis symmetrical	x = sin(u).[exp(cos(u)) – 2.cos(4u) – sin(u/12)^5] y = cos(u).[exp(cos(u)) – 2.cos(4u) – sin(u/12)^5] Parameter range u[0.0: 2.0 PI]	1800	10875	7348	7345	3527 32 %

SUMMARY OF RESULTS

BUTTERFLY CURVE - FEEDRATES
Fig 1 – Butterfly curve. Algo 1. The x-feedrate and y-feedrate satisfy the feedrate limit. But the feedrate limit is jagged and not smooth.
Fig 2 – Butterfly curve. Algo 2. The x-feedrate and y-feedrate satisfy the feedrate limit. Chord segments are small. Increase interpolated points.
Fig 3 – Butterfly curve. Algo 3. The x-feedrate and y-feedrate satisfy the feedrate limit. Contour error is constrained. Smoother feedrates.
Fig 4 – Butterfly curve. Algo 4 (last step). The x-feedrate and y-feedrate satisfy the feedrate limit. Contour error constrained. Smoother feedrates.

BUTTERFLY CURVE - CHORD ERRORS
Fig 5 - Algo 1 -Butterfly curve error profile (eps). The error tolerance is 0.000001 or 10(-6). The error (eps) is much higher than tolerance for all u.
Fig 6 - Algo2 - Butterfly curve error profile (eps). The error tolerance is below 0.000001 or 10(-6) for all values of parameter u. The very small values of eps (below the lowest line, which are outside of this figure) increases the number of interpolated points. Small eps means small chord errors. With smaller chord errors (eps) means more interpolated points.
Fig 7 - .Algo 3 - Butterfly curve error profile (eps). Again the error tolerance is below 0.000001 or 10(-6) for all values of parameter u. Bringing up the errors closer to tolerance significantly decreases the number of interpolated points since the chord length increases.
Fig 8 - Algo 4 -Butterfly curve error profile (eps). The error tolerance is maintained below 10(-6) for all values of parameter u. Raising the errors further is time consuming since it requires a lot of computation time. There is no significant improvement in decreasing the number of interpolated points. However, there is significant improvement in reducing the tangential acceleration to remain between the (min, max) limits.

Fig 9 - The computing environment with live data display running Butterfly curve.
Fig 10 - Running simulation of the NGC Code for the Butterfly curve.

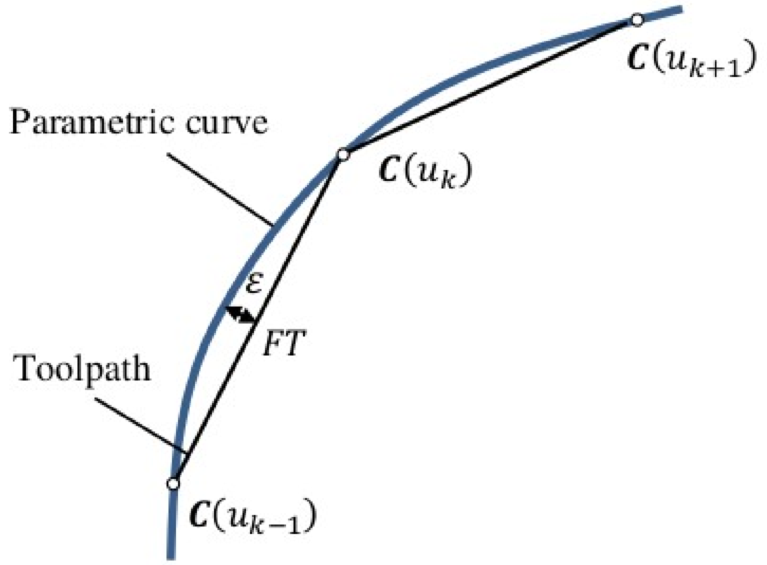
FEEDRATES
Fig 11 Run Algo 1 for the Snailshell curve
Fig 12 Run Algo 2 for the Snailshell curve
Fig 13 Run Algo 3 for the Snailshell curve

FEEDRATES
Fig 14 Run Algo 1 for the Hypotrocoid curve.
Fig 15 Run Algo 2 for the Hypotrocoid curve.
Fig 16 Run Algo 3 for the Hypotrocoid curve.

FINDING FEEDRATE LIMITS
Fig - 17 Butterfly. Algo 2. Finding the feedrate limit. (Minimum graph)
Fig - 18 Butterfly. Algo 3. Finding the feedrate limit. (Minimum graph is now smooth)
Fig - 19 Butterfly. Algo 4. Finding the feedrate limit. (Not much different from Algo 3)

TANGENTIAL ACCELERATIONS
Fig -20 Butterfly. Algo 2. The calculated tangential acceleration.
Fig -21 Butterfly. Algo 3. The calculated tangential acceleration.
Fig -22 Butterfly. Algo 4. The calculated tangential acceleration. It is now reduced throughout all values of u.

The chord error epsilon is shown in the figure below.



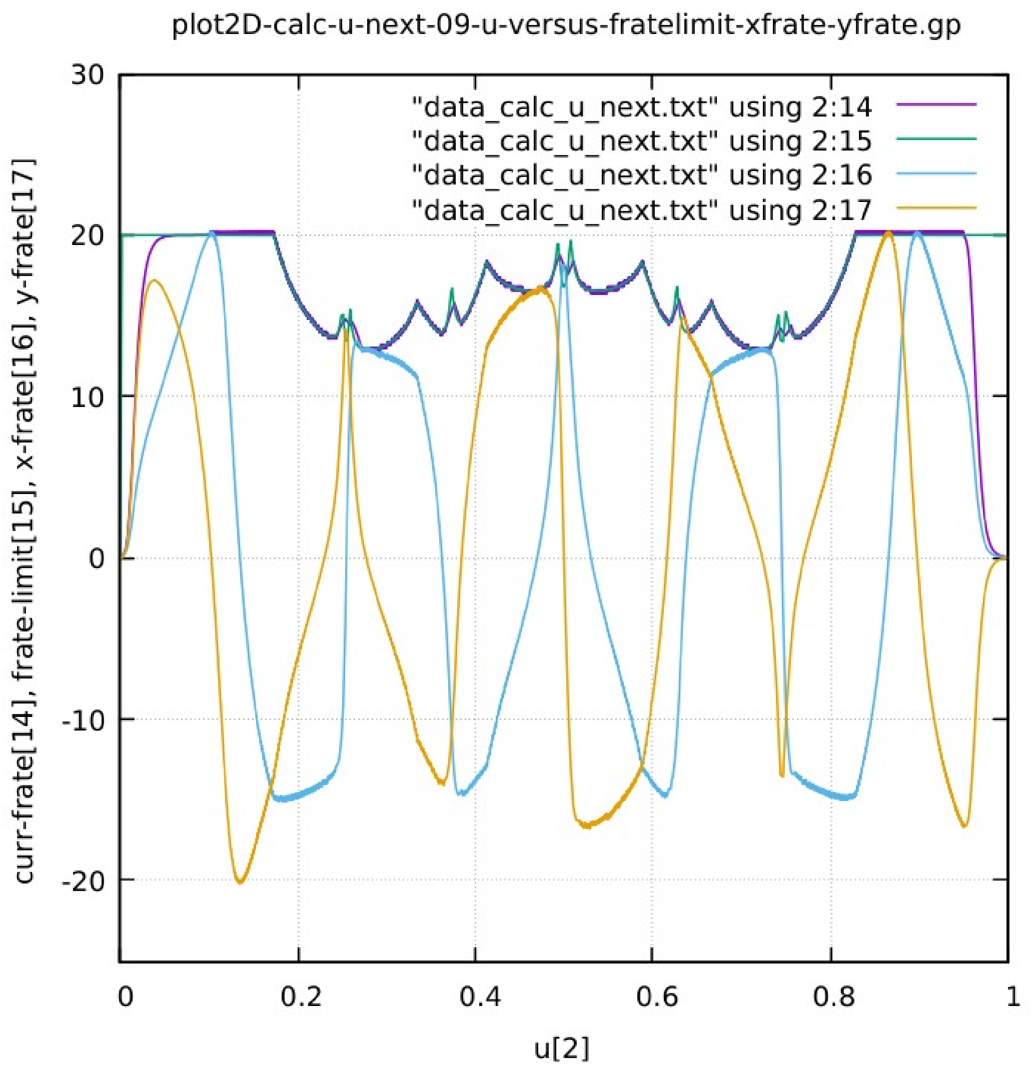


Fig 1 – Butterfly curve. Algo 1. The x-feedrate and y-feedrate satisfy the feedrate limit. But the feedrate limit is jagged and not smooth.

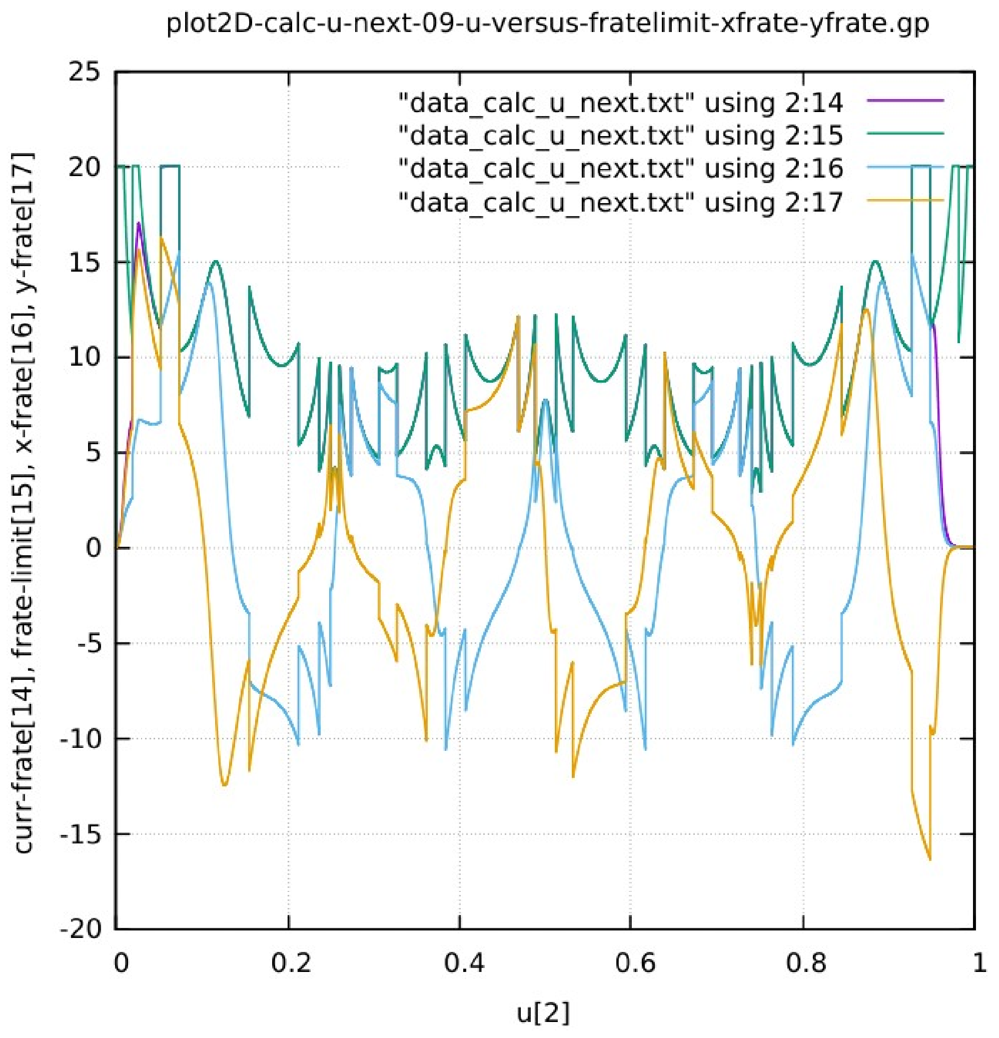


Fig 2 – Butterfly curve. Algo 2. The x-feedrate and y-feedrate satisfy the feedrate limit. Chord segments are small. Increase interpolated points.

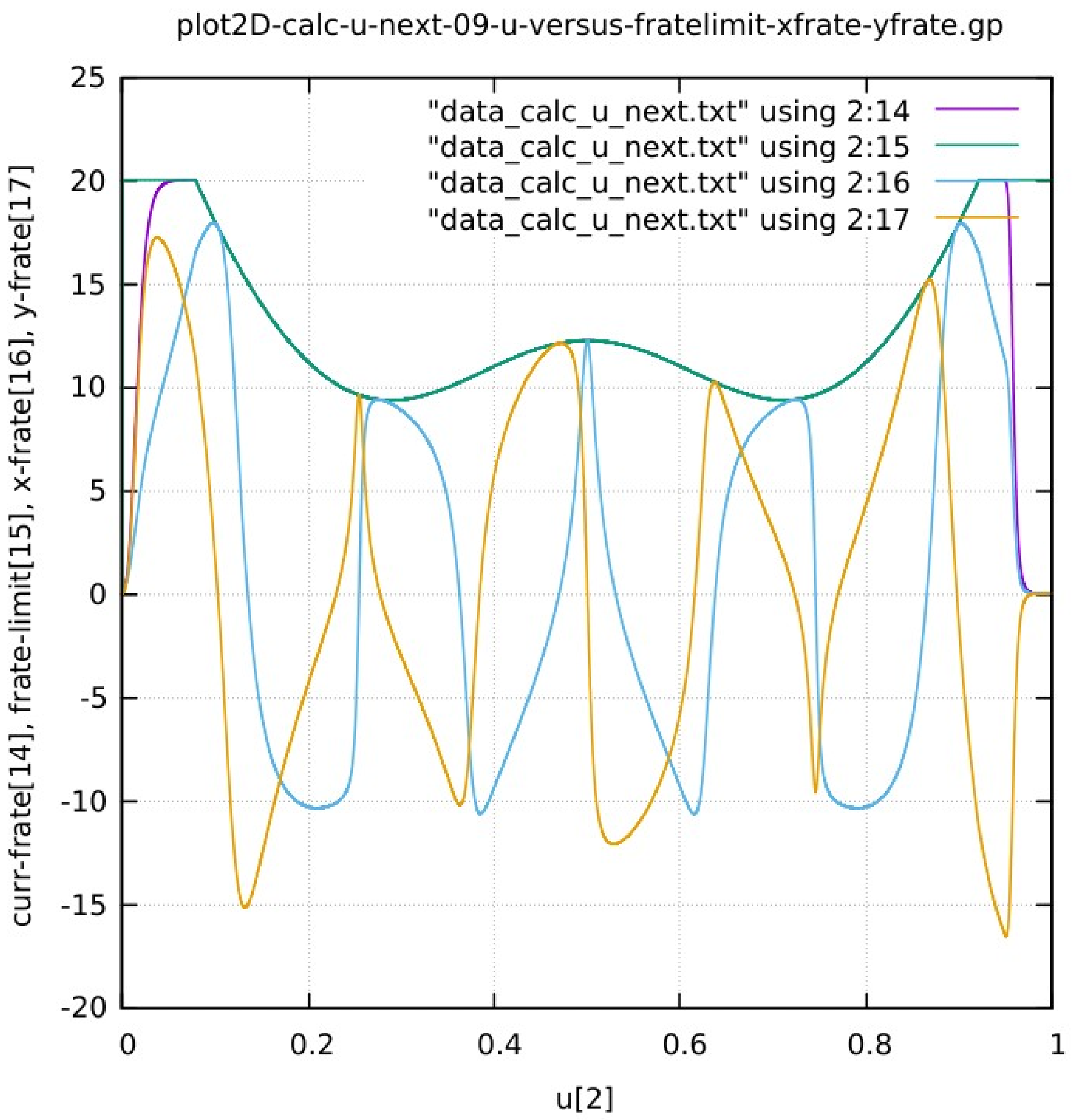


Fig 3 – Butterfly curve. Algo 3 The x-feedrate and y-feedrate satisfy the feedrate limit. Contour error is constrained. Smoother feedrates.

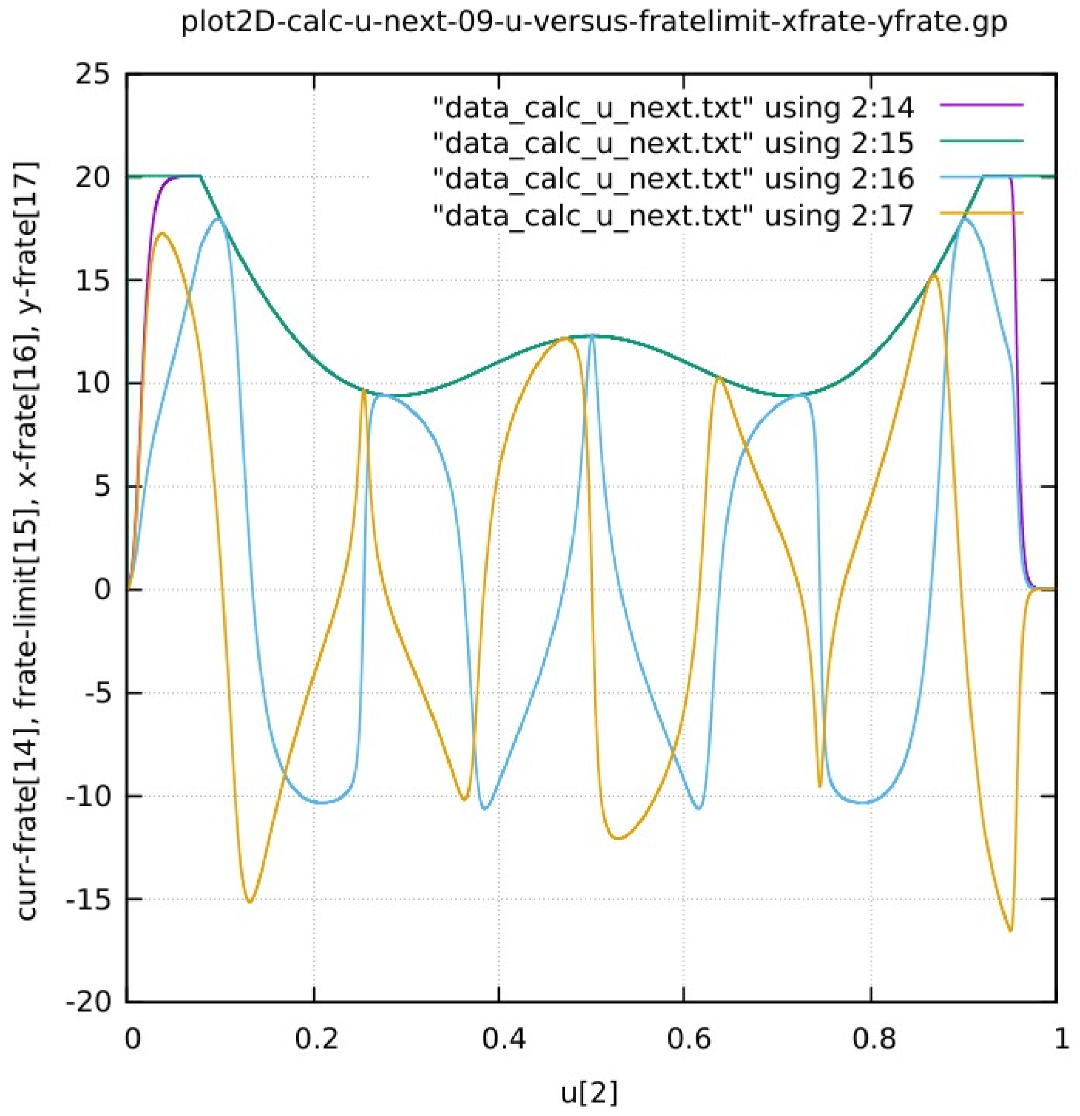


Fig 4– Butterfly curve. Algo 4 (last step). The x-feedrate and y-feedrate satisfy the feedrate limit. Contour error constraint. Smoother feedrates.

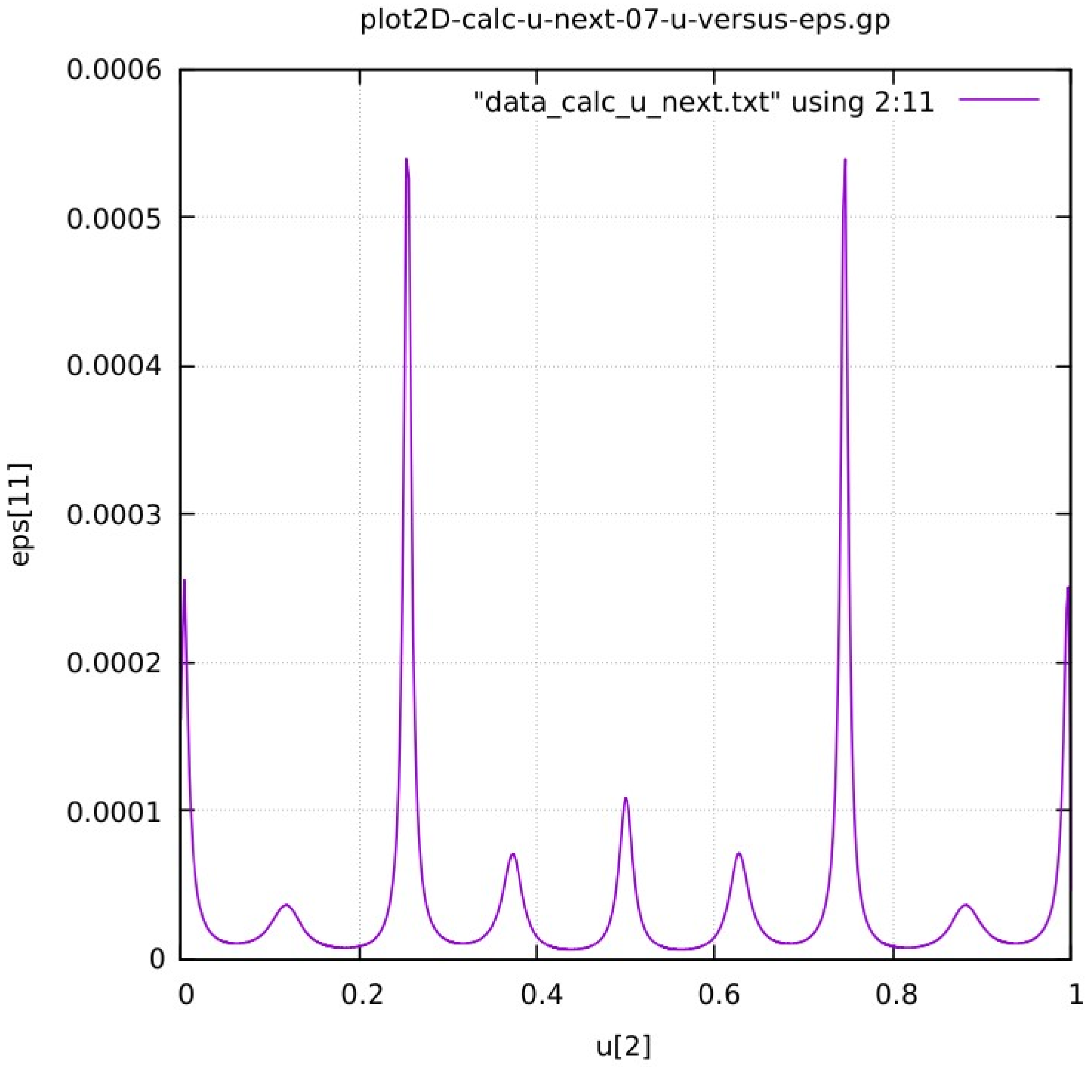


Fig 5 - Butterfly curve Algo 1 - Error profile (eps). The error tolerance is 0.000001 or 10(-6). Error (eps) is much higher than tolerance for all u. The graph above does mot show 0.000001 on the y-axis. The y-axis scale is 10(-4) while the tolerance is 10(-6).

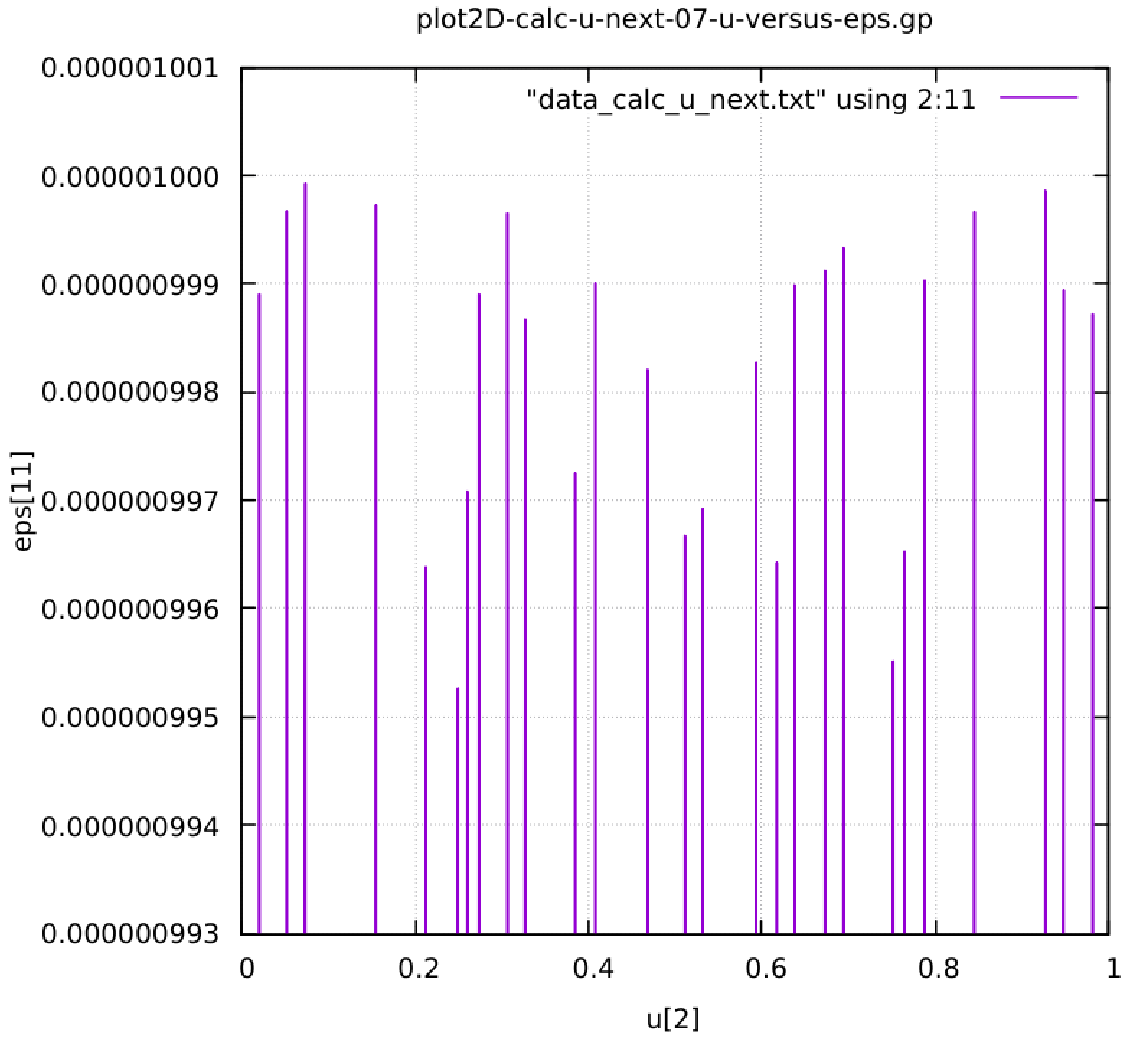


Fig 6 Butterfly curve Algo 2. Error profile (eps). The error tolerance is below 0.000001 or 10(-6) for all values of parameter u. The very small values of eps (below the lowest line, which are outside of this figure) increases the number of interpolated points. Small eps means small chord errors. With smaller chord errors (eps) means more interpolated points.

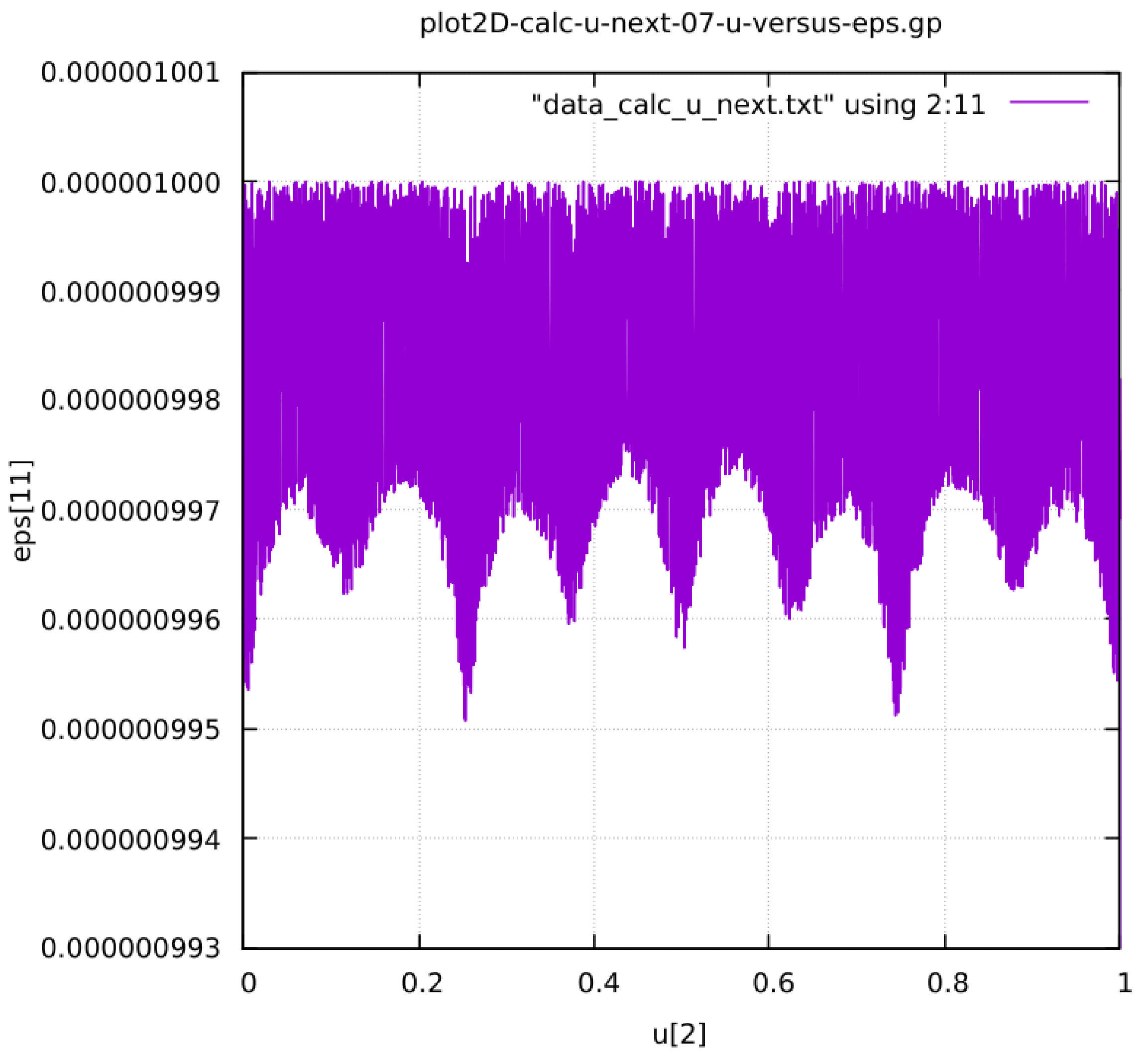


Fig 7 Algo 3 - Butterfly curve error profile (eps). Again the error tolerance is below 0.000001 or 10(-6) for all values of parameter u. Bringing up the errors closer to tolerance significantly decreases the number of interpolated points since the chord length increases.

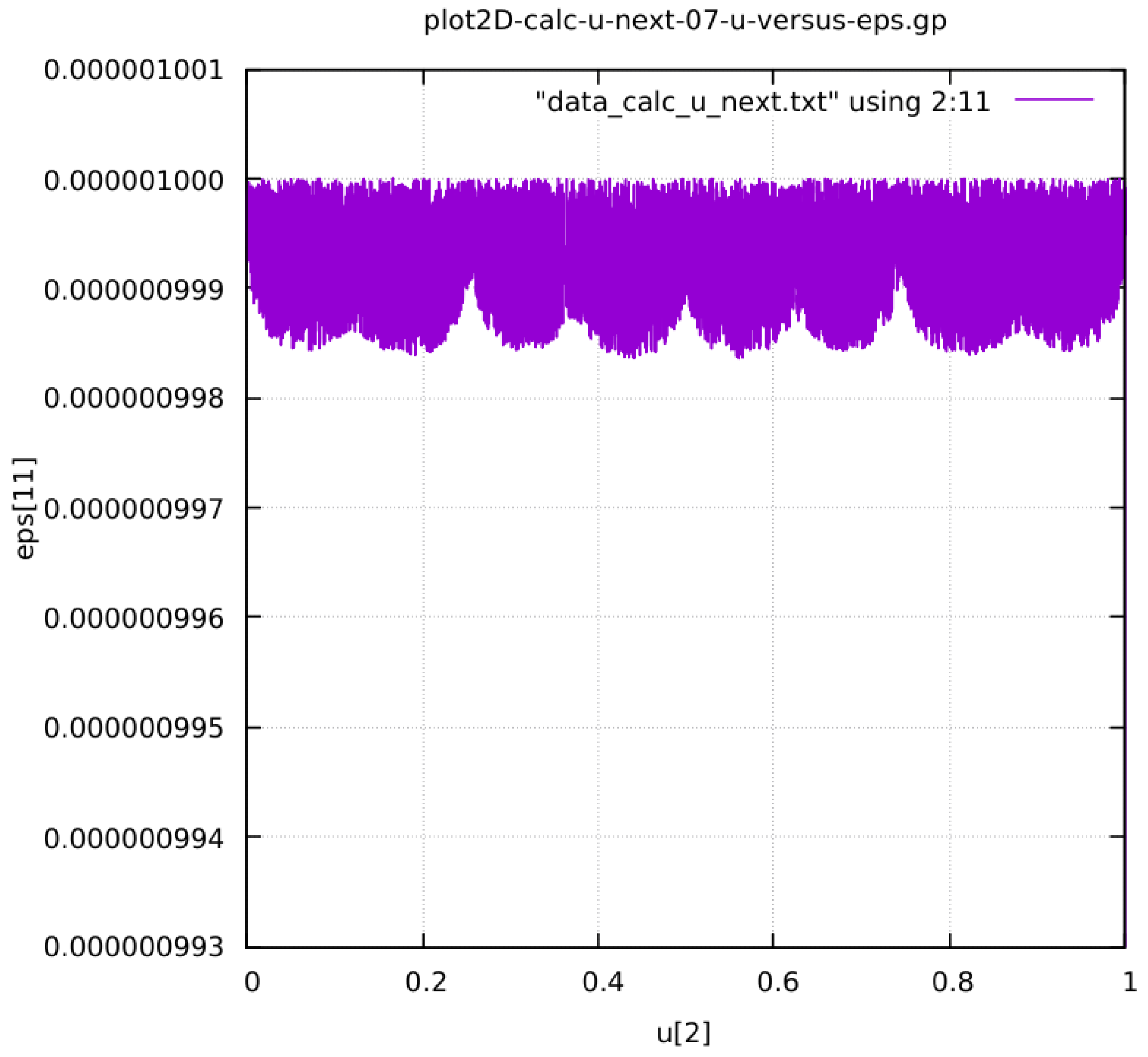
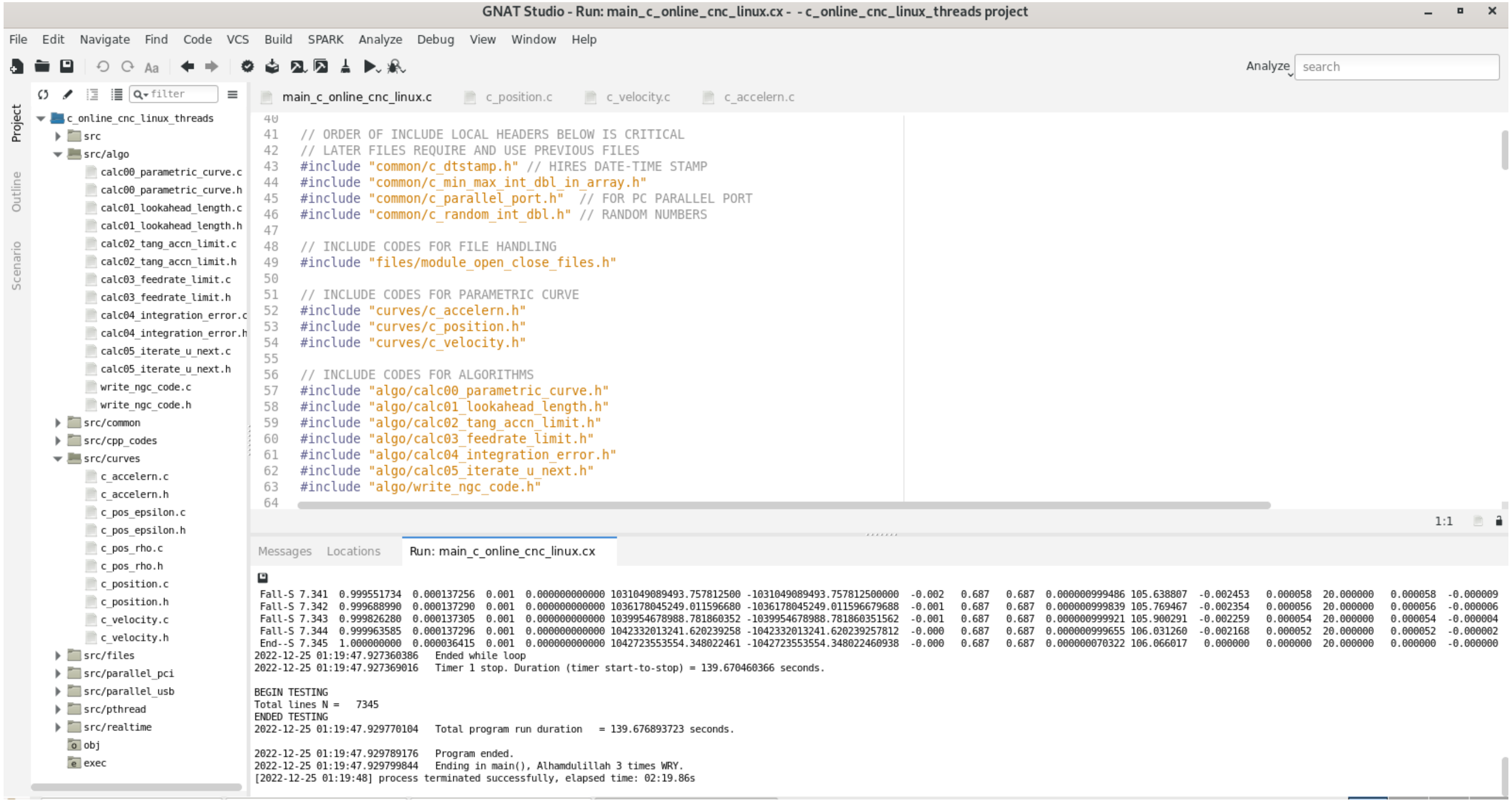


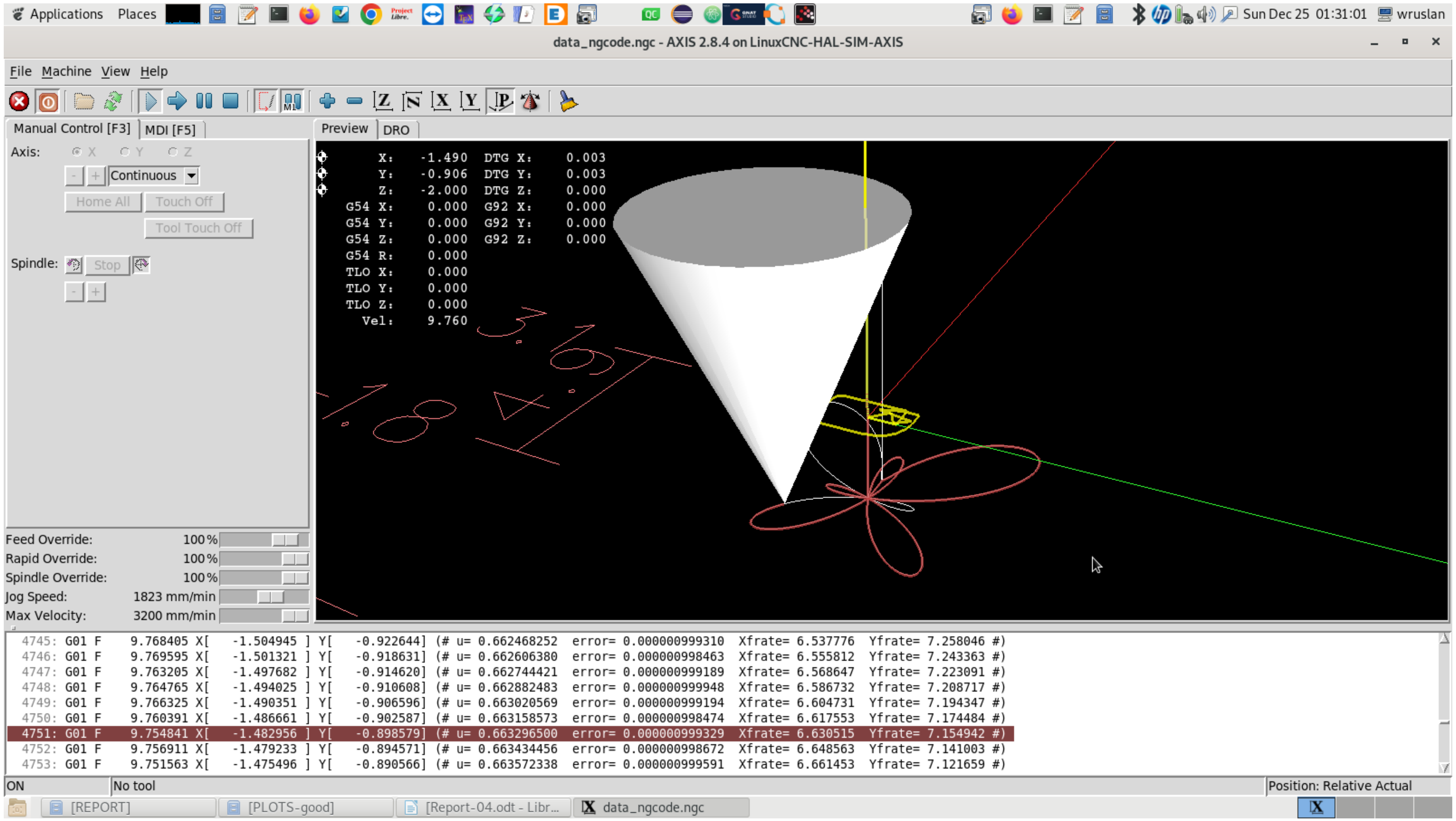
Fig 8 Algo 4 -Butterfly curve error profile (eps). The error tolerance is maintained below 10(-6) for all values of parameter u. Raising the errors further is time consuming since it requires a lot of computation time. There is no significant improvement in decreasing the number of interpolated points. However, there is significant improvement in reducing the tangential acceleration to remain between the (min, max) limits.



```c
// ORDER OF INCLUDE LOCAL HEADERS BELOW IS CRITICAL
// LATER FILES REQUIRE AND USE PREVIOUS FILES
#include "common/c_dtstamp.h" // HIRES DATE-TIME STAMP
#include "common/c_min_max_int_dbl_in_array.h"
#include "common/c_parallel_port.h"  // FOR PC PARALLEL PORT
#include "common/c_random_int_dbl.h" // RANDOM NUMBERS

// INCLUDE CODES FOR FILE HANDLING
#include "files/module_open_close_files.h"

// INCLUDE CODES FOR PARAMETRIC CURVE
#include "curves/c_accelern.h"
#include "curves/c_position.h"
#include "curves/c_velocity.h"

// INCLUDE CODES FOR ALGORITHMS
#include "algo/calc00_parametric_curve.h"
#include "algo/calc01_lookahead_length.h"
#include "algo/calc02_tang_accn_limit.h"
#include "algo/calc03_feedrate_limit.h"
#include "algo/calc04_integration_error.h"
#include "algo/calc05_iterate_u_next.h"
#include "algo/write_ngc_code.h"
```

```
 Fall-S 7.341  0.999551734  0.000137256  0.001  0.000000000000 1031049089493.757812500 -1031049089493.757812500000  -0.002   0.687   0.687  0.000000999486 105.638807  -0.002453   0.000058  20.000000   0.000058  -0.000009
 Fall-S 7.342  0.999688990  0.000137290  0.001  0.000000000000 1036178045249.011596680 -1036178045249.011596679688  -0.001   0.687   0.687  0.000000999839 105.769467  -0.002354   0.000056  20.000000   0.000056  -0.000006
 Fall-S 7.343  0.999826280  0.000137305  0.001  0.000000000000 1039954678988.781860352 -1039954678988.781860351562  -0.001   0.687   0.687  0.000000999921 105.900291  -0.002259   0.000054  20.000000   0.000054  -0.000004
 Fall-S 7.344  0.999963585  0.000137296  0.001  0.000000000000 1042332013241.620239258 -1042332013241.620239257812  -0.000   0.687   0.687  0.000000999655 106.031260  -0.002168   0.000052  20.000000   0.000052  -0.000002
 End--S 7.345  1.000000000  0.000036415  0.001  0.000000000000 1042723553554.348022461 -1042723553554.348022460938  -0.000   0.687   0.687  0.000000070322 106.066017   0.000000   0.000000  20.000000   0.000000  -0.000000
2022-12-25 01:19:47.927360386   Ended while loop
2022-12-25 01:19:47.927369016   Timer 1 stop. Duration (timer start-to-stop) = 139.670460366 seconds.

BEGIN TESTING
Total lines N =   7345
ENDED TESTING
2022-12-25 01:19:47.929770104   Total program run duration   = 139.676893723 seconds.

2022-12-25 01:19:47.929789176   Program ended.
2022-12-25 01:19:47.929799844   Ending in main(), Alhamdulillah 3 times WRY.
[2022-12-25 01:19:48] process terminated successfully, elapsed time: 02:19.86s
```

Fig-9 The computing environment with live data display.



```
      X:  -1.490  DTG X:   0.003
      Y:  -0.906  DTG Y:   0.003
      Z:  -2.000  DTG Z:   0.000
  G54 X:   0.000  G92 X:   0.000
  G54 Y:   0.000  G92 Y:   0.000
  G54 Z:   0.000  G92 Z:   0.000
  G54 R:   0.000
  TLO X:   0.000
  TLO Y:   0.000
  TLO Z:   0.000
    Vel:   9.760
```

```
Feed Override:        100 %
Rapid Override:       100 %
Spindle Override:     100 %
Jog Speed:       1823 mm/min
Max Velocity:    3200 mm/min
```

```
G01 F    9.768405 X[   -1.504945 ] Y[   -0.922644] (# u= 0.662468252  error= 0.000000999310  Xfrate= 6.537776  Yfrate= 7.258046 #)
G01 F    9.769595 X[   -1.501321 ] Y[   -0.918631] (# u= 0.662606380  error= 0.000000998463  Xfrate= 6.555812  Yfrate= 7.243363 #)
G01 F    9.763205 X[   -1.497682 ] Y[   -0.914620] (# u= 0.662744421  error= 0.000000999189  Xfrate= 6.568647  Yfrate= 7.223091 #)
G01 F    9.764765 X[   -1.494025 ] Y[   -0.910608] (# u= 0.662882483  error= 0.000000999948  Xfrate= 6.586732  Yfrate= 7.208717 #)
G01 F    9.766325 X[   -1.490351 ] Y[   -0.906596] (# u= 0.663020569  error= 0.000000999194  Xfrate= 6.604731  Yfrate= 7.194347 #)
G01 F    9.760391 X[   -1.486661 ] Y[   -0.902587] (# u= 0.663158573  error= 0.000000998474  Xfrate= 6.617553  Yfrate= 7.174484 #)
G01 F    9.754841 X[   -1.482956 ] Y[   -0.898579] (# u= 0.663296500  error= 0.000000999329  Xfrate= 6.630515  Yfrate= 7.154942 #)
G01 F    9.756911 X[   -1.479233 ] Y[   -0.894571] (# u= 0.663434456  error= 0.000000998672  Xfrate= 6.648563  Yfrate= 7.141003 #)
G01 F    9.751563 X[   -1.475496 ] Y[   -0.890566] (# u= 0.663572338  error= 0.000000999591  Xfrate= 6.661453  Yfrate= 7.121659 #)
```

```
ON    No tool    Position: Relative Actual
```

Fig 10 – Running simulation of the Butterfly NGC Code.

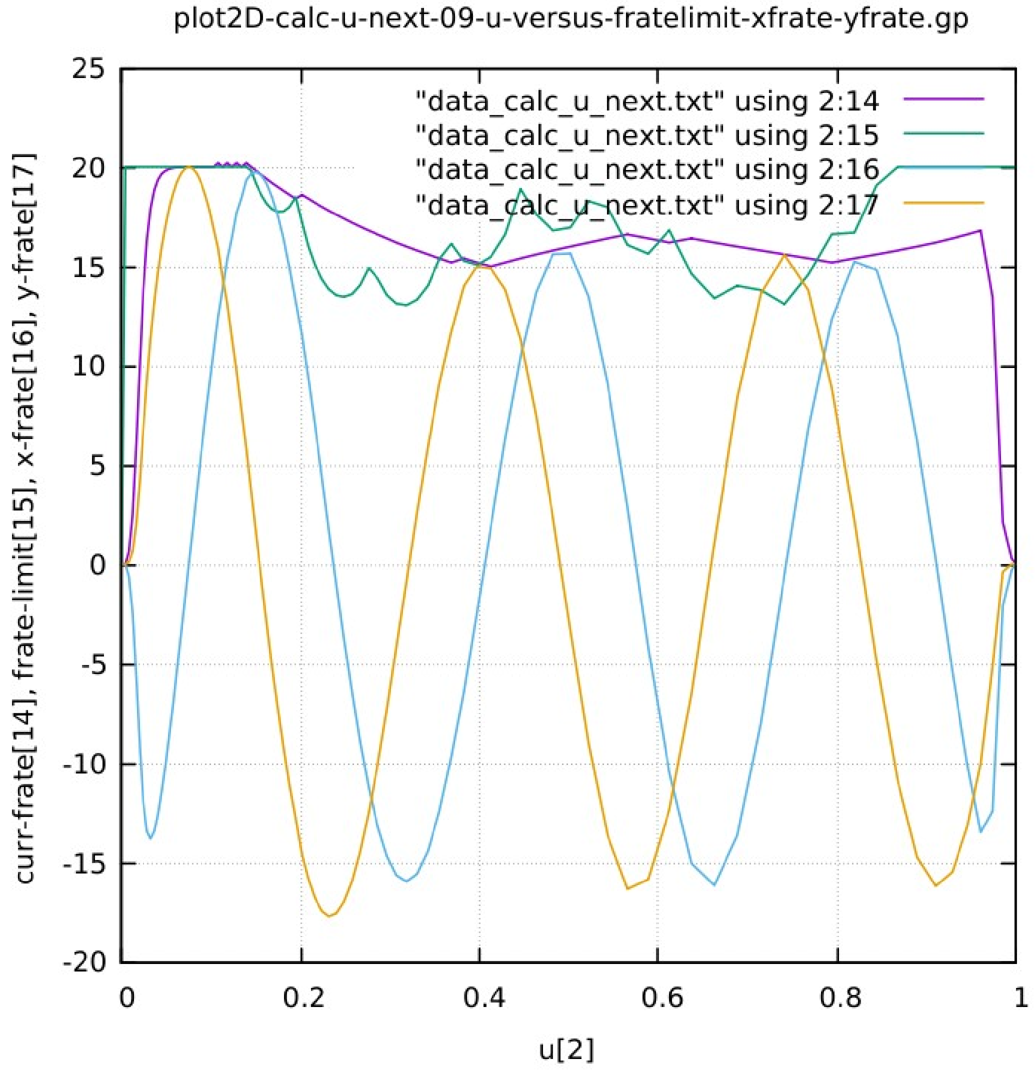


Fig -11 Run Algo 1 for the Snailshell

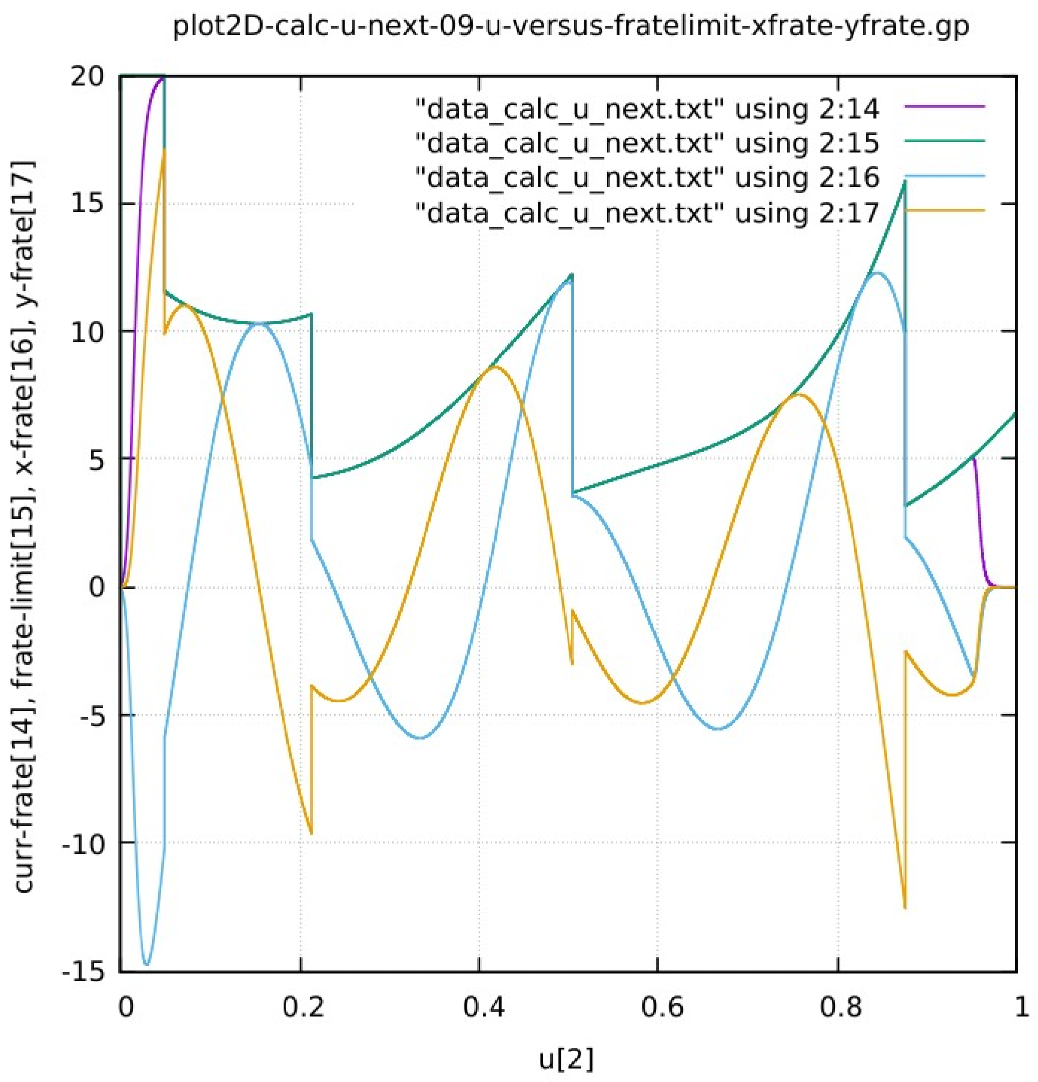


Fig-12 Run Algo 2 for Snailshell.

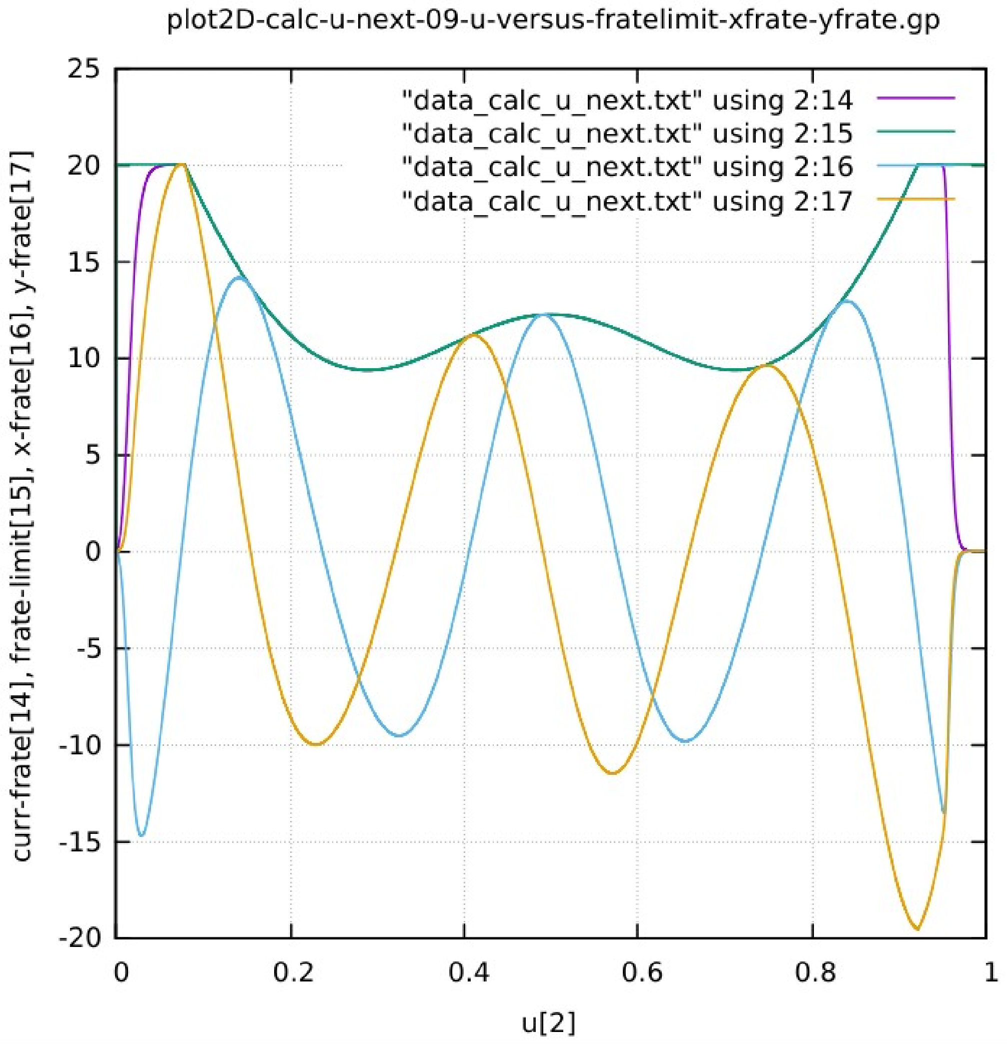


Fig 13 Run Algo 3 for Snailshell.

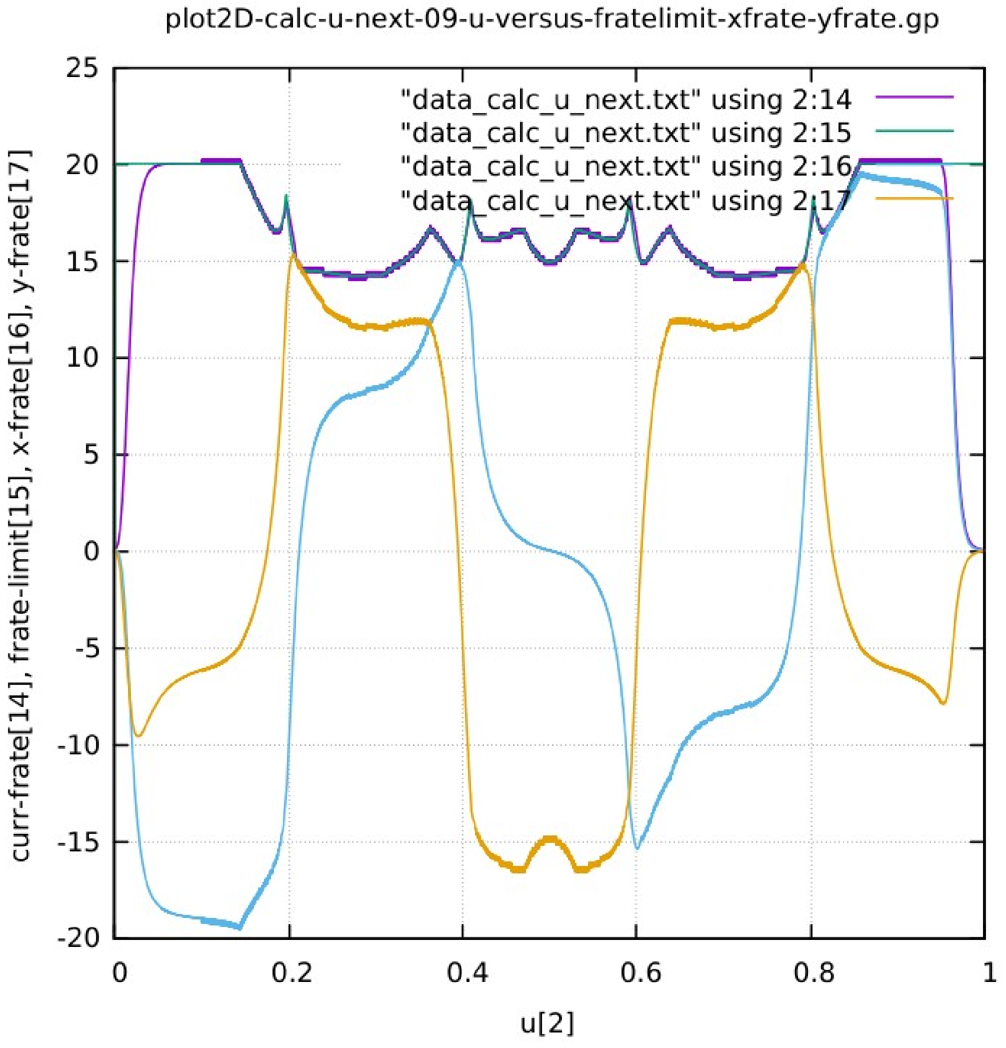


Fig 14 Run Algo 1 for the Hypotrocoid curve.

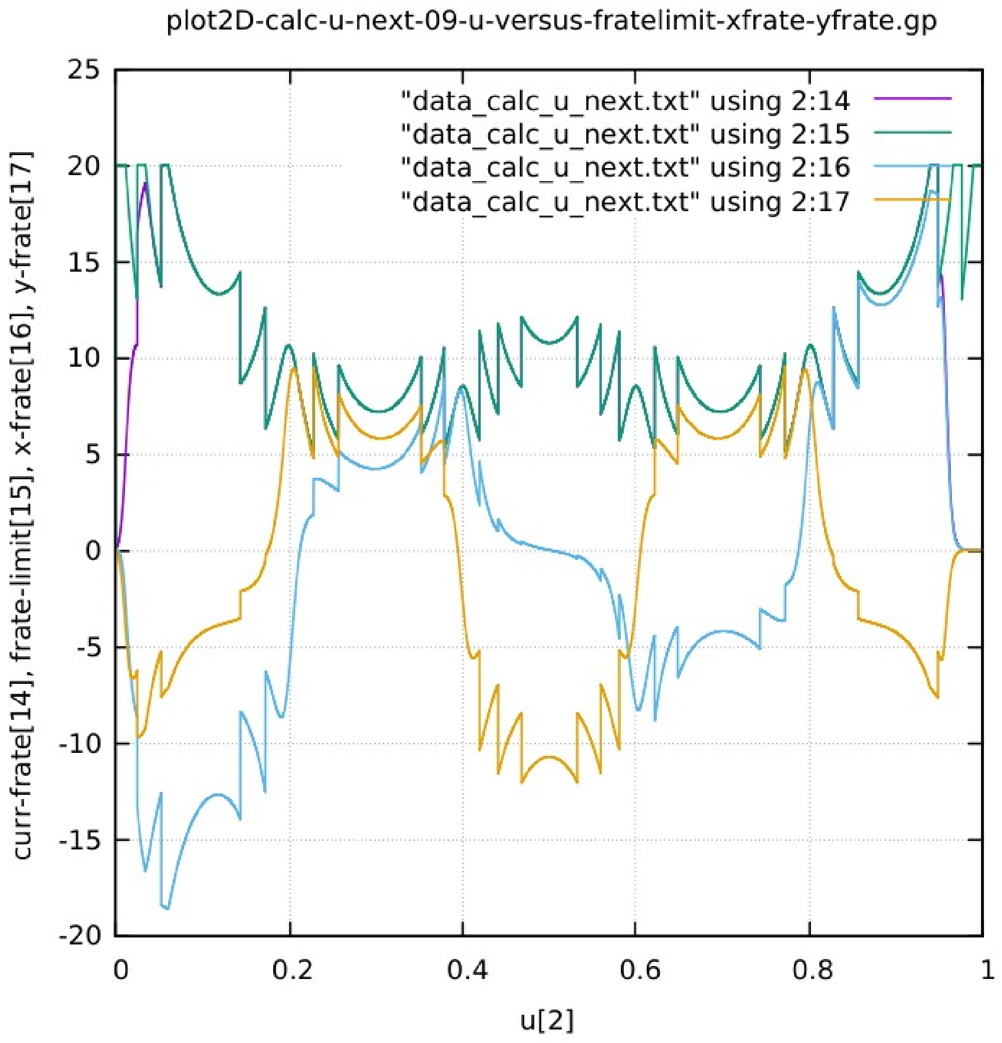


Fig 15 Run Algo 2 for the Hypotrocoid curve.

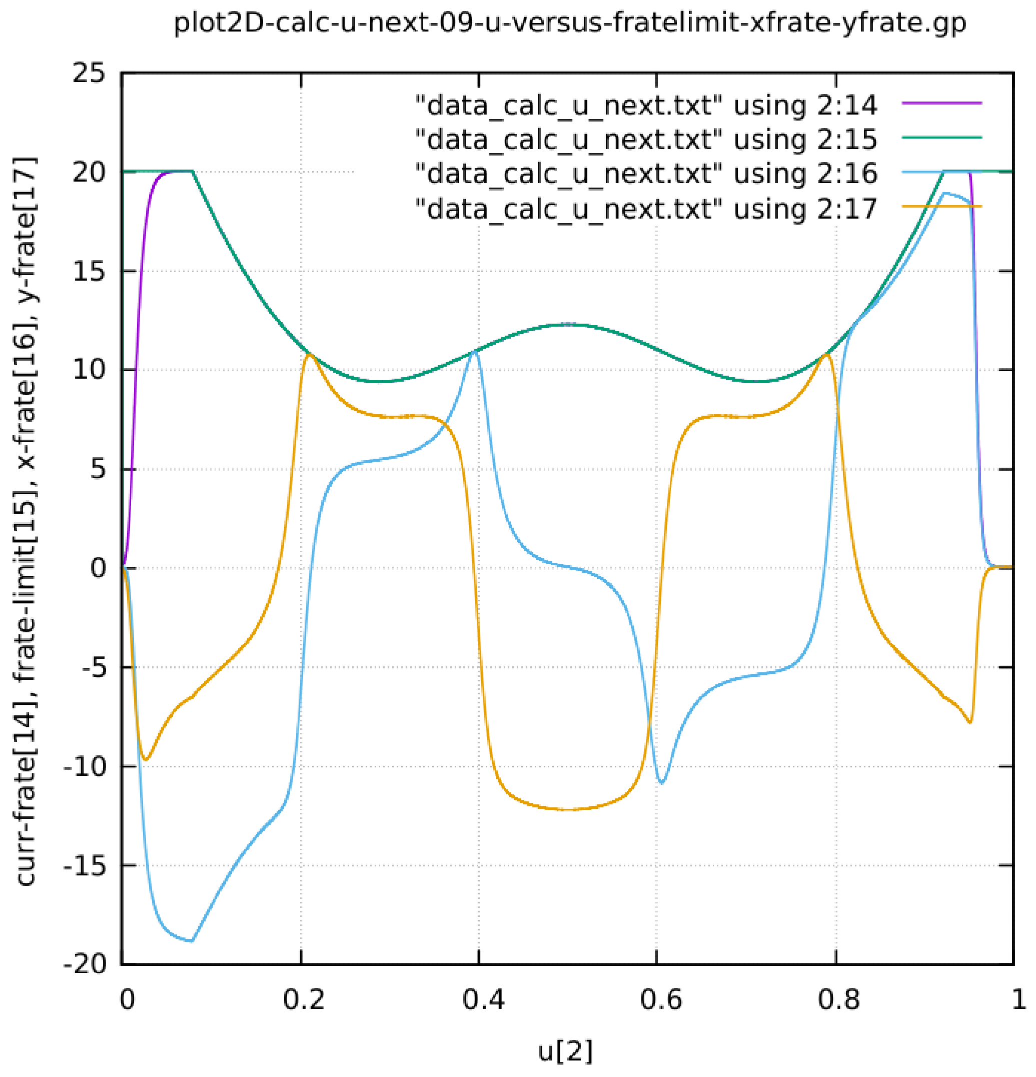


Fig 16 Run Algo 3 for the Hypotrocoid curve.

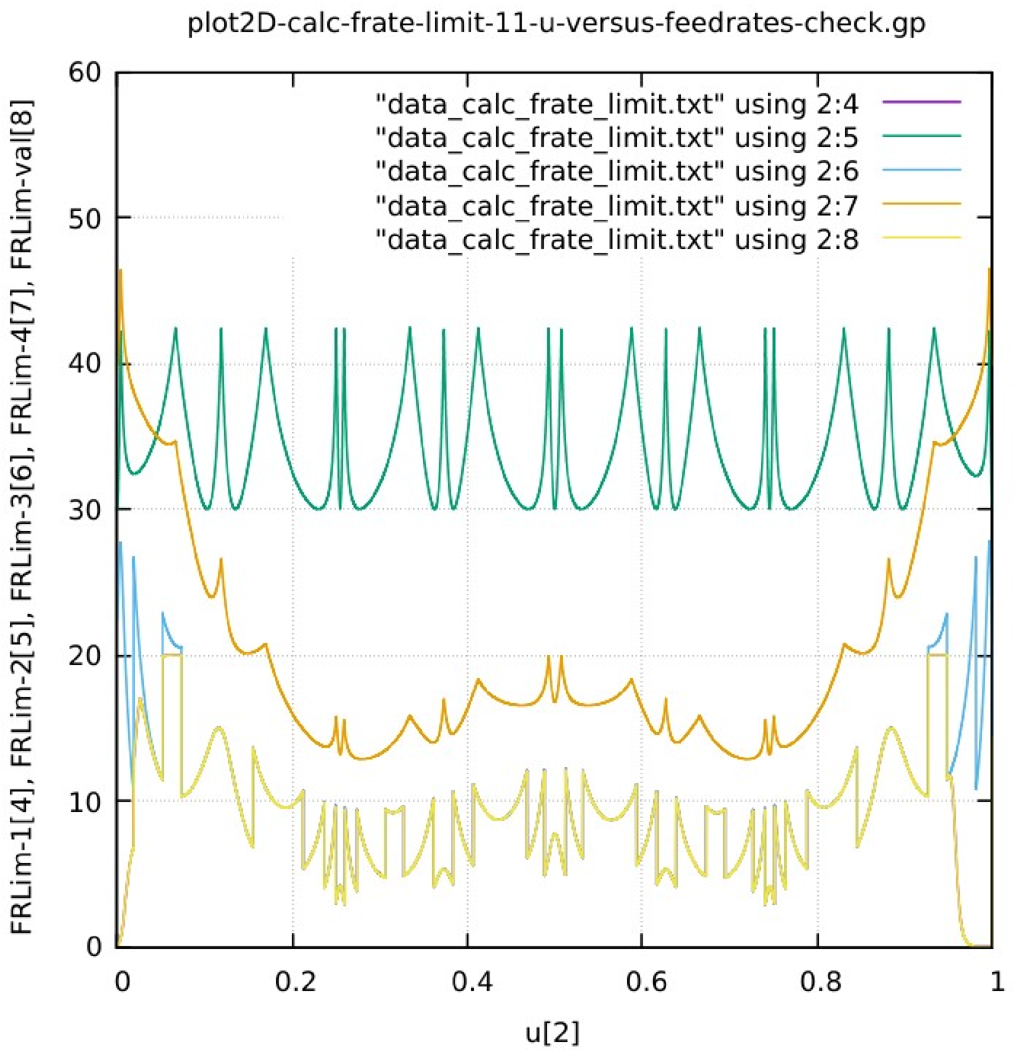


Fig -17 Butterfly. Algo 2. Finding the feedrate limit. (Minimum graph)

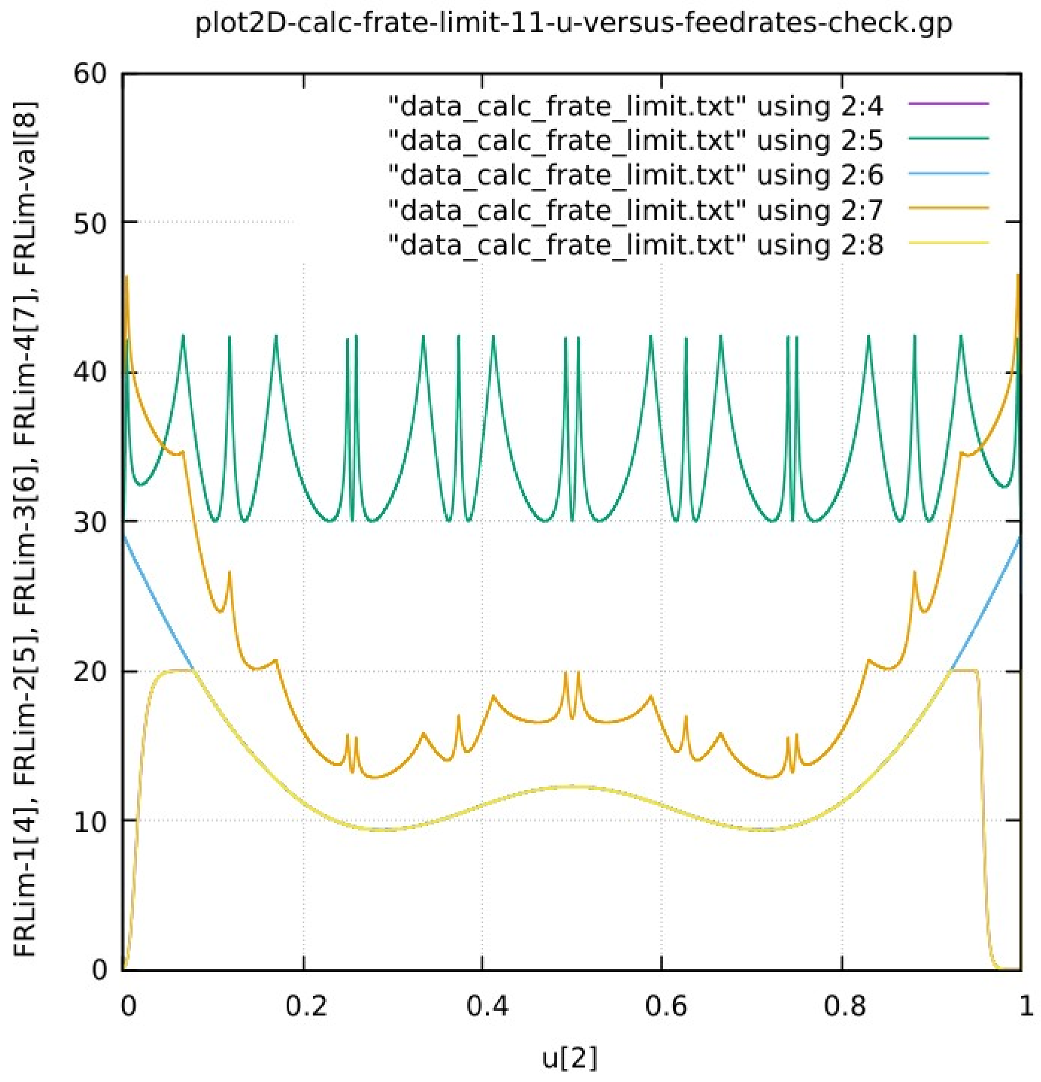


Fig 18 - Butterfly. Algo 3. Finding the feedrate limit. (Minimum graph is now smooth)

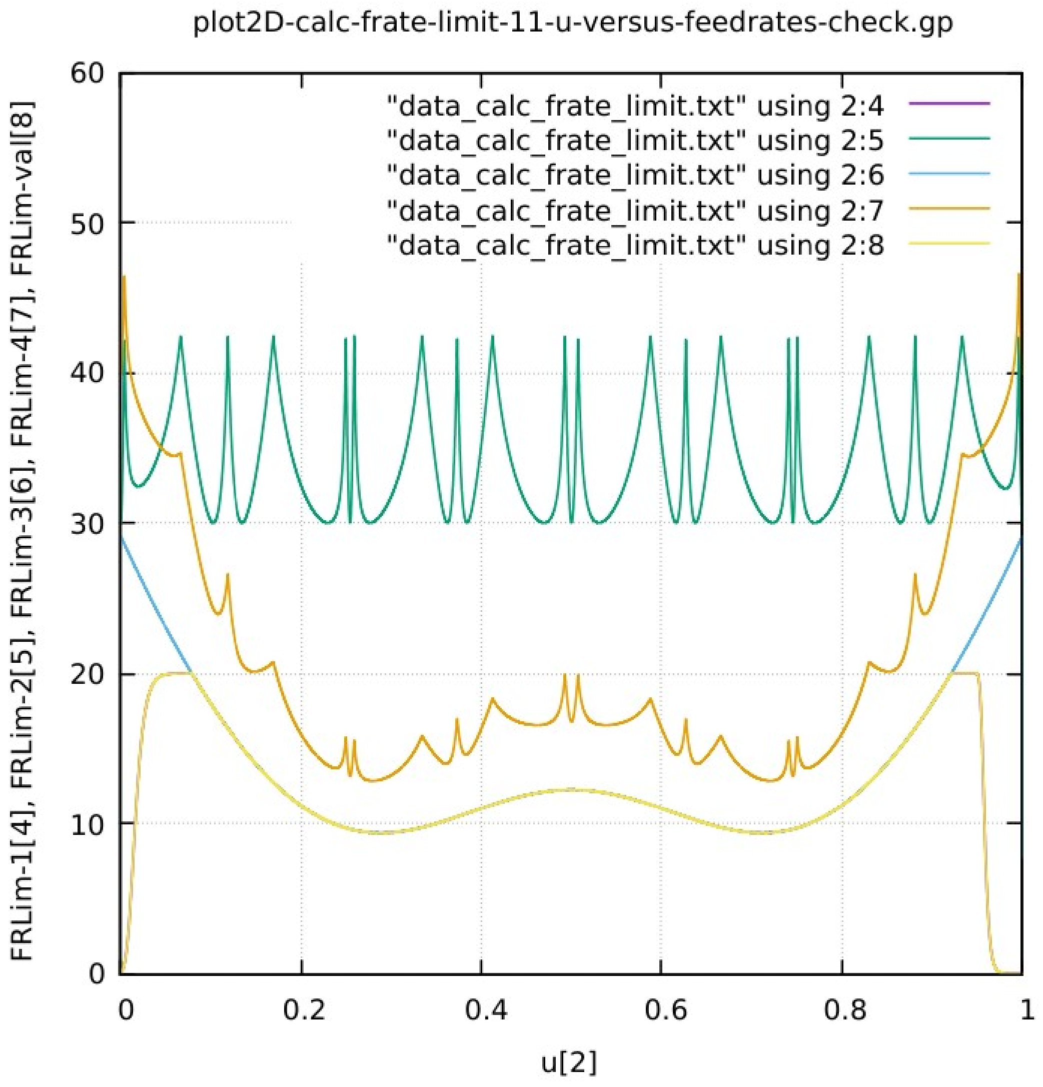


Fig 19 - Butterfly. Algo 4. Finding the feedrate limit. (Not much different from Algo 3).

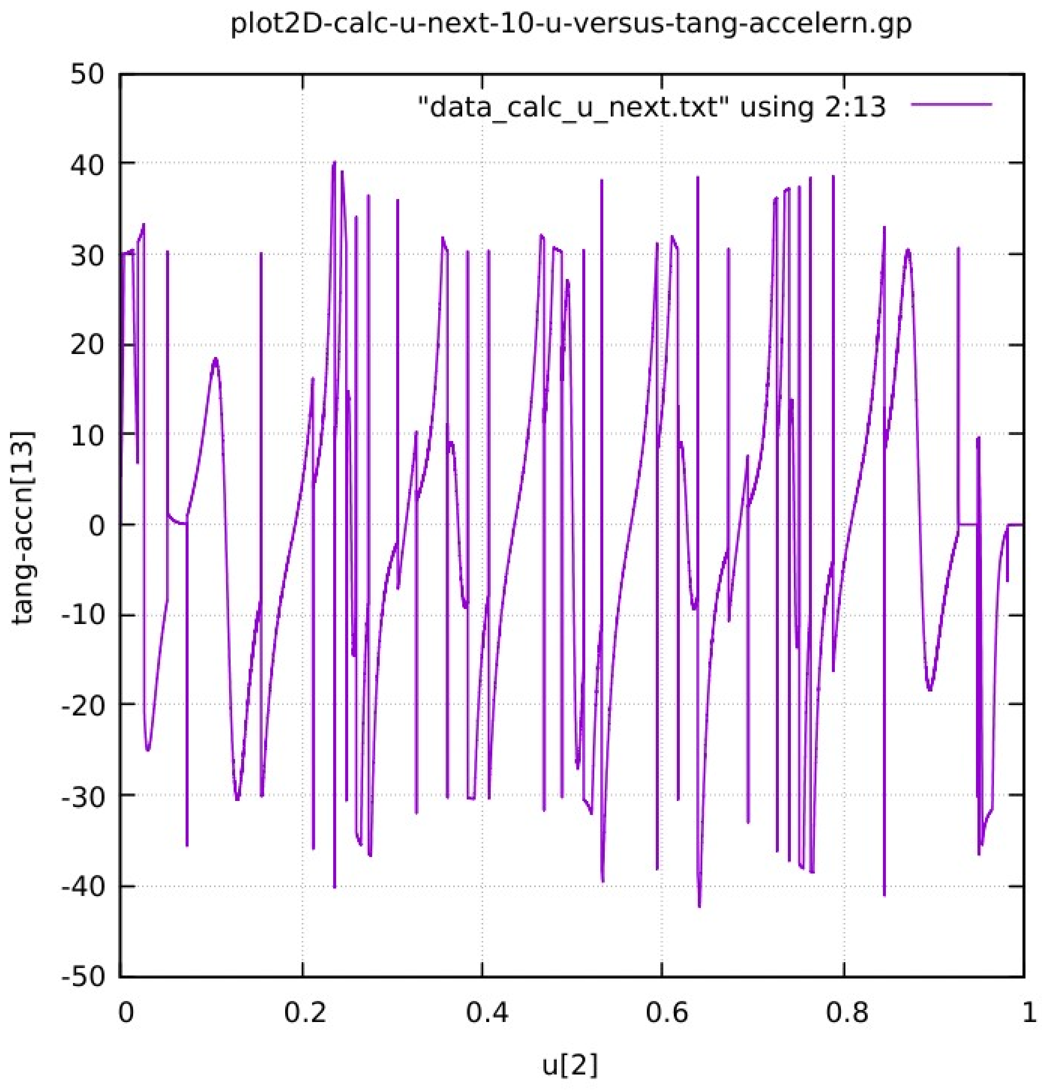


Fig - 20 Butterfly. Algo 2. The calculated tangential acceleration.

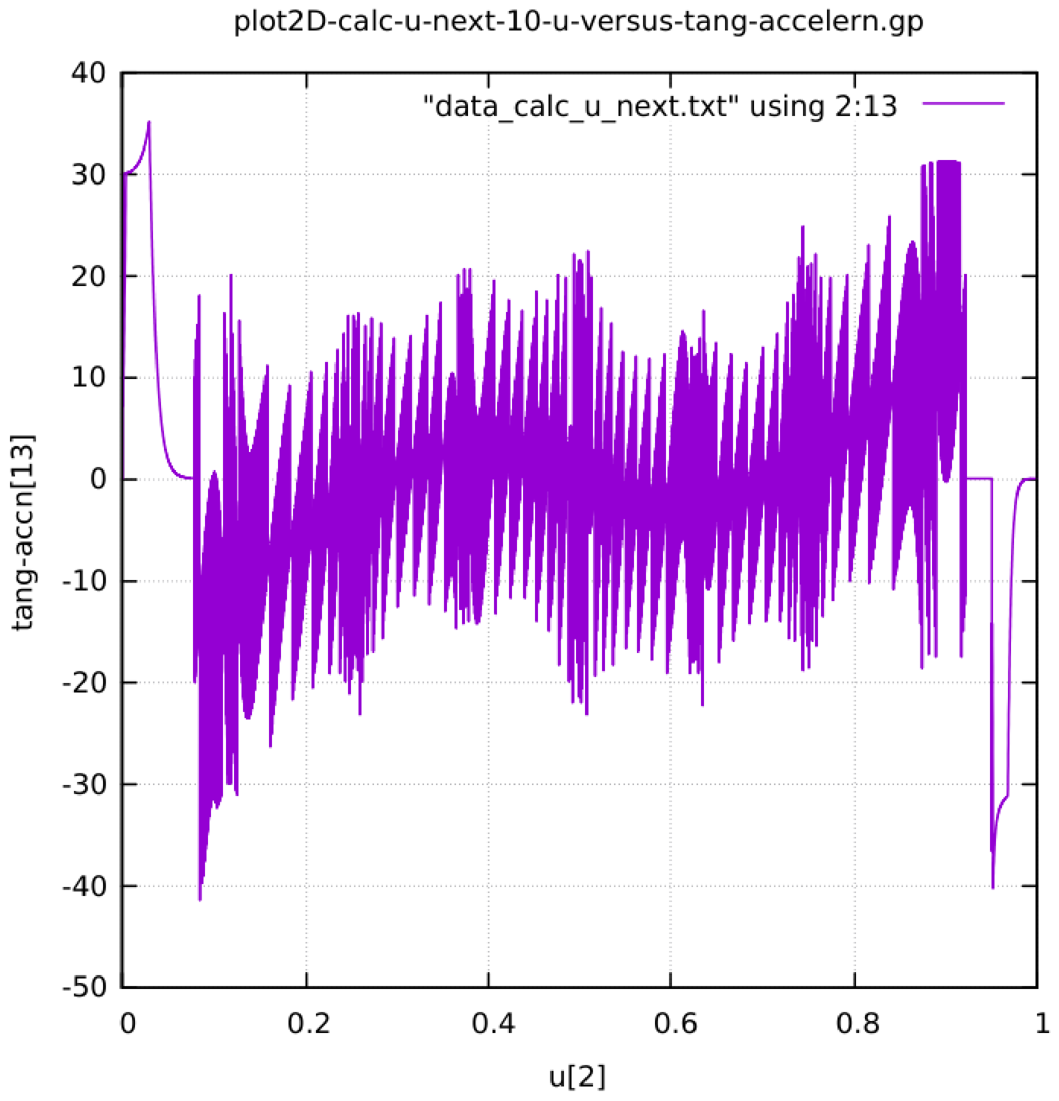


Fig -21 Butterfly. Algo 3. The calculated tangential acceleration.

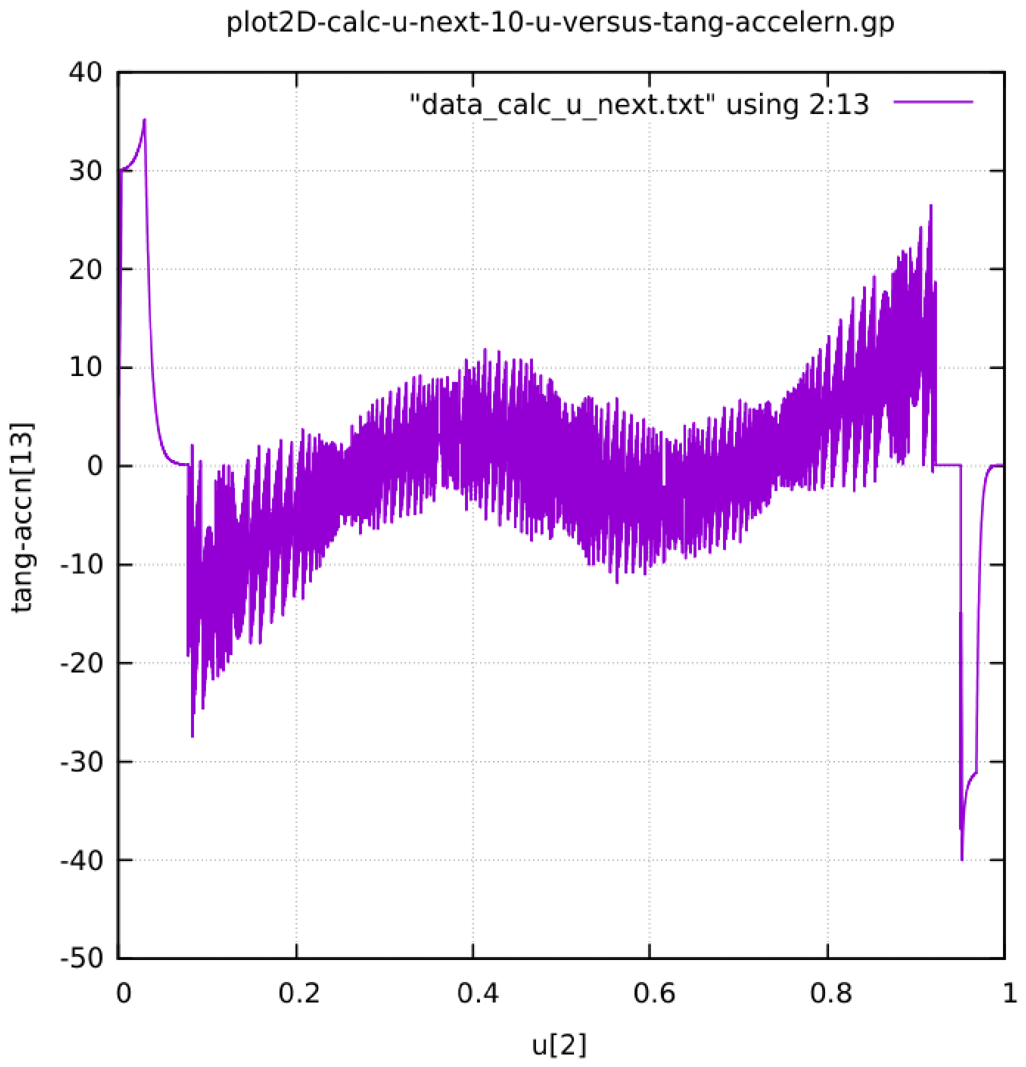


Fig -22 Butterfly. Algo 4. The calculated tangential acceleration. It is now reduced throughout.

CONCLUSIONS:

This study introduces a constraint on the chord error that significantly decreases the number of interpolated points by about 20%. The less number of interpolated points means increase productivity, since there are less total points to process..

The imposition of a maximum chord error constraint, for example 0.000001 or 10(-6), increases chord error, but the trade-off is an increase in feedrate smoothness (curvature) profiles and thus, results in overall reduction of machine jerk and acceleration, within its respective minimum and maximum limits.

The parametric curve as input must be continuous, including its first and second derivatives. This ensures that there are no phenomena like cusps or discontinuous jumps in the first and second derivatives.. The presence of discontinuity causes a circular repetition in interpolated points and the interpolator algorithm will not exit.

Parametric curves that are very small in magnitude (overall size of the x-y curve) may also cause a similar circular repetition in interpolated points with the interpolator not exiting. The solution is to scale up the parametric curve. The cause of this interpolation behaviour in the algorithm is due to the finite limit of the smallest number that can be represented by the computing system.

Wassalam.
Wan Ruslan Yusoff.