

The Hobby package: code

Andrew Stacey

loopspace@mathforge.org

1.10 from 2023-08-30

1 Implementation

1.1 Main Code

We use L^AT_EX3 syntax so need to load the requisite packages

```
1 \RequirePackage{pml3array}
2 \ExplSyntaxOn

3 \cs_generate_variant:Nn \fp_set:Nn {Nx}
4 \cs_generate_variant:Nn \tl_if_eq:nnTF {VnTF}
5 \cs_generate_variant:Nn \tl_if_eq:nnTF {xnTF}
```

1.1.1 Initialisation

We declare all our variables.

Start with version and date, together with a check to see if we've been loaded twice (fail gracefully if so).

```
6 \tl_clear:N \l_tmpa_tl
7 \tl_if_exist:NT \g__hobby_version
8 {
9   \tl_set:Nn \l_tmpa_tl {
10     \ExplSyntaxOff
11     \tl_clear:N \l_tmpa_tl
12     \endinput
13   }
14 }
15 \tl_use:N \l_tmpa_tl
16
17 \tl_new:N \g__hobby_version
18 \tl_new:N \g__hobby_date
19 \tl_gset:Nn \g__hobby_version {1.10}
20 \tl_gset:Nn \g__hobby_date {2023-08-30}
21 \DeclareDocumentCommand \hobbyVersion {}
22 {
23   \tl_use:N \g__hobby_version
24 }
25 \DeclareDocumentCommand \hobbyDate {}
26 {
27   \tl_use:N \g__hobby_date
28 }
```

The function for computing the lengths of the control points depends on three parameters. These are set to $a = \sqrt{2}$, $b = 1/16$, and $c = \frac{3-\sqrt{5}}{2}$.

```
29 \fp_new:N \g_hobby_parama_fp
30 \fp_new:N \g_hobby_paramb_fp
31 \fp_new:N \g_hobby_paramc_fp
32 \fp_gset:Nn \g_hobby_parama_fp {2^.5}
```

```

33 \fp_gset:Nn \g_hobby_paramb_fp {1/16}
34 \fp_gset:Nn \g_hobby_paramc_fp {(3-5^5)/2}

```

Now we define our objects for use in generating the path.

`\l_hobby_closed_bool` `\l_hobby_closed_bool` is true if the path is closed.

```

35 \bool_new:N \l_hobby_closed_bool

```

(End of definition for \l_hobby_closed_bool. This function is documented on page ??.)

`\l_hobby_disjoint_bool` `\l_hobby_disjoint_bool` is true if the path should start with a `moveto` command.

```

36 \bool_new:N \l_hobby_disjoint_bool

```

(End of definition for \l_hobby_disjoint_bool. This function is documented on page ??.)

`\l_hobby_save_aux_bool` `\l_hobby_save_aux_bool` is true if when saving paths then they should be saved to the aux file.

```

37 \bool_new:N \l_hobby_save_aux_bool
38 \bool_set_true:N \l_hobby_save_aux_bool
39 \DeclareDocumentCommand \HobbyDisableAux {}
40 {
41   \bool_set_false:N \l_hobby_save_aux_bool
42 }

```

(End of definition for \l_hobby_save_aux_bool. This function is documented on page ??.)

`\g__hobby_points_array` `\g__hobby_points_array` is an array holding the specified points on the path. In the L^AT_EX3 code, a “point” is a token list of the form `<number>`, `<number>`. This gives us the greatest flexibility in passing points back and forth between the L^AT_EX3 code and any calling code. The array is indexed by integers beginning with 0. In the documentation, we will use the notation z_k to refer to the k th point.

```

43 \array_new:N \g__hobby_points_array

```

(End of definition for \g__hobby_points_array.)

`\g__hobby_points_x_array` `\g__hobby_points_x_array` is an array holding the x -coordinates of the specified points.

```

44 \array_new:N \g__hobby_points_x_array

```

(End of definition for \g__hobby_points_x_array.)

`\g__hobby_points_y_array` `\g__hobby_points_y_array` is an array holding the y -coordinates of the specified points.

```

45 \array_new:N \g__hobby_points_y_array

```

(End of definition for \g__hobby_points_y_array. This function is documented on page ??.)

`\g__hobby_actions_array` `\g__hobby_actions_array` is an array holding the (encoded) action to be taken out on the segment of the path ending at that point.

```

46 \array_new:N \g__hobby_actions_array

```

(End of definition for \g__hobby_actions_array.)

`\g__hobby_angles_array` `\g__hobby_angles_array` is an array holding the angles of the lines between the points. Specifically, the angle indexed by k is the angle in radians of the line from z_k to z_{k+1} .

```

47 \array_new:N \g__hobby_angles_array

```

(End of definition for \g__hobby_angles_array.)

`\g__hobby_distances_array` `\g__hobby_distances_array` is an array holding the distances between the points. Specifically, the distance indexed by k , which we will write as d_k , is the length of the line from z_k to z_{k+1} .

```

48 \array_new:N \g__hobby_distances_array

```

(End of definition for \g__hobby_distances_array.)

`\g__hobby_tension_out_array` `\g__hobby_tension_out_array` is an array holding the tension for the path as it leaves each point. This is a parameter that controls how much the curve “flexes” as it leaves the point. In the following, this will be written τ_k .

```
49 \array_new:N \g__hobby_tension_out_array
```

(End of definition for `\g__hobby_tension_out_array`.)

`\g__hobby_tension_in_array` `\g__hobby_tension_in_array` is an array holding the tension for the path as it arrives at each point. This is a parameter that controls how much the curve “flexes” as it gets to the point. In the following, this will be written $\bar{\tau}_k$.

```
50 \array_new:N \g__hobby_tension_in_array
```

(End of definition for `\g__hobby_tension_in_array`.)

`\g__hobby_matrix_a_array` `\g__hobby_matrix_a_array` is an array holding the subdiagonal of the linear system that has to be solved to find the angles of the control points. In the following, this will be denoted by A_i . The first index is 1.

```
51 \array_new:N \g__hobby_matrix_a_array
```

(End of definition for `\g__hobby_matrix_a_array`.)

`\g__hobby_matrix_b_array` `\g__hobby_matrix_b_array` is an array holding the diagonal of the linear system that has to be solved to find the angles of the control points. In the following, this will be denoted by B_i . The first index is 0.

```
52 \array_new:N \g__hobby_matrix_b_array
```

(End of definition for `\g__hobby_matrix_b_array`.)

`\g__hobby_matrix_c_array` `\g__hobby_matrix_c_array` is an array holding the superdiagonal of the linear system that has to be solved to find the angles of the control points. In the following, this will be denoted by C_i . The first index is 0.

```
53 \array_new:N \g__hobby_matrix_c_array
```

(End of definition for `\g__hobby_matrix_c_array`.)

`\g__hobby_matrix_d_array` `\g__hobby_matrix_d_array` is an array holding the target vector of the linear system that has to be solved to find the angles of the control points. In the following, this will be denoted by D_i . The first index is 1.

```
54 \array_new:N \g__hobby_matrix_d_array
```

(End of definition for `\g__hobby_matrix_d_array`.)

`\g__hobby_vector_u_array` `\g__hobby_vector_u_array` is an array holding the perturbation of the linear system for closed paths. The coefficient matrix for an *open* path is tridiagonal and that means that Gaussian elimination runs faster than expected ($O(n)$ instead of $O(n^3)$). The matrix for a closed path is not tridiagonal but is not far off. It can be solved by perturbing it to a tridiagonal matrix and then modifying the result. This array represents a utility vector in that perturbation. In the following, the vector will be denoted by u . The first index is 1.

```
55 \array_new:N \g__hobby_vector_u_array
```

(End of definition for `\g__hobby_vector_u_array`.)

`\g__hobby_excess_angle_array` `\g__hobby_excess_angle_array` is an array that allows the user to say that the algorithm should add a multiple of 2π to the angle differences. This is because these angles are wrapped to the interval $(-\pi, \pi]$ but the wrapping might go wrong near the end points due to computation accuracy. The first index is 1.

```
56 \array_new:N \g__hobby_excess_angle_array
```

(End of definition for `\g__hobby_excess_angle_array`.)

`\g__hobby_psi_array` `\g__hobby_psi_array` is an array holding the difference of the angles of the lines entering and exiting a point. That is, ψ_k is the angle between the lines joining z_k to z_{k-1} and z_{k+1} . The first index is 1.

```
57 \array_new:N \g__hobby_psi_array
```

(End of definition for `\g__hobby_psi_array`.)

`\g__hobby_theta_array` `\g__hobby_theta_array` is an array holding the angles of the outgoing control points for the generated path. These are measured relative to the line joining the point to the next point on the path. The first index is 0.

```
58 \array_new:N \g__hobby_theta_array
```

(End of definition for `\g__hobby_theta_array`.)

`\g__hobby_phi_array` `\g__hobby_phi_array` is an array holding the angles of the incoming control points for the generated path. These are measured relative to the line joining the point to the previous point on the path. The first index is 1.

```
59 \array_new:N \g__hobby_phi_array
```

(End of definition for `\g__hobby_phi_array`.)

`\g__hobby_sigma_array` `\g__hobby_sigma_array` is an array holding the lengths of the outgoing control points for the generated path. The units are such that the length of the line to the next specified point is one unit.

```
60 \array_new:N \g__hobby_sigma_array
```

(End of definition for `\g__hobby_sigma_array`.)

`\g__hobby_rho_array` `\g__hobby_rho_array` is an array holding the lengths of the incoming control points for the generated path. The units are such that the length of the line to the previous specified point is one unit.

```
61 \array_new:N \g__hobby_rho_array
```

(End of definition for `\g__hobby_rho_array`.)

`\g__hobby_controla_array` `\g__hobby_controla_array` is an array holding the coordinates of the first control points on the curves. The format is the same as for `\g__hobby_points_array`.

```
62 \array_new:N \g__hobby_controla_array
```

(End of definition for `\g__hobby_controla_array`.)

`\g__hobby_controlb_array` `\g__hobby_controlb_array` is an array holding the coordinates of the second control points on the curves. The format is the same as for `\g__hobby_points_array`.

```
63 \array_new:N \g__hobby_controlb_array
```

(End of definition for `\g__hobby_controlb_array`.)

`\l_hobby_matrix_v_fp` `\l_hobby_matrix_v_fp` is a number which is used when doing the perturbation of the solution of the linear system for a closed curve. There is actually a vector, v , that this corresponds to but that vector only has one component that needs computation.

```
64 \fp_new:N \l_hobby_matrix_v_fp
```

(End of definition for `\l_hobby_matrix_v_fp`. This function is documented on page ??.)

`\l_hobby_tempa_tl` `\l_hobby_tempa_tl` is a temporary variable of type `tl`.

```
65 \fp_new:N \l_hobby_tempa_tl
```

(End of definition for `\l_hobby_tempa_tl`. This function is documented on page ??.)

`\l_hobby_tempb_tl` `\l_hobby_tempb_tl` is a temporary variable of type `tl`.

```
66 \fp_new:N \l_hobby_tempb_tl
```

(End of definition for \l_hobby_tempb_tl. This function is documented on page ??.)

`\l_hobby_tempa_fp` `\l_hobby_tempa_fp` is a temporary variable of type `fp`.

`67 \fp_new:N \l_hobby_tempa_fp`

(End of definition for \l_hobby_tempa_fp. This function is documented on page ??.)

`\l_hobby_tempb_fp` `\l_hobby_tempb_fp` is a temporary variable of type `fp`.

`68 \fp_new:N \l_hobby_tempb_fp`

(End of definition for \l_hobby_tempb_fp. This function is documented on page ??.)

`\l_hobby_tempc_fp` `\l_hobby_tempc_fp` is a temporary variable of type `fp`.

`69 \fp_new:N \l_hobby_tempc_fp`

(End of definition for \l_hobby_tempc_fp. This function is documented on page ??.)

`\l_hobby_tempd_fp` `\l_hobby_tempd_fp` is a temporary variable of type `fp`.

`70 \fp_new:N \l_hobby_tempd_fp`

(End of definition for \l_hobby_tempd_fp. This function is documented on page ??.)

`\l_hobby_temps_fp` `\l_hobby_temps_fp` is a temporary variable of type `fp`.

`71 \fp_new:N \l_hobby_temps_fp`

(End of definition for \l_hobby_temps_fp. This function is documented on page ??.)

`\g__hobby_in_curl_fp` `\g__hobby_in_curl_fp` is the “curl” at the end of an open path. This is used if the angle at the end is not specified.

`72 \fp_new:N \g__hobby_in_curl_fp`

`73 \fp_gset:Nn \g__hobby_in_curl_fp {1}`

(End of definition for \g__hobby_in_curl_fp.)

`\g__hobby_out_curl_fp` `\g__hobby_out_curl_fp` is the “curl” at the start of an open path. This is used if the angle at the start is not specified.

`74 \fp_new:N \g__hobby_out_curl_fp`

`75 \fp_gset:Nn \g__hobby_out_curl_fp {1}`

(End of definition for \g__hobby_out_curl_fp.)

`\g__hobby_in_angle_fp` `\g__hobby_in_angle_fp` is the angle at the end of an open path. If this is not specified, it will be computed automatically. It is set to `\c_inf_fp` to allow easy detection of when it has been specified.

`76 \fp_new:N \g__hobby_in_angle_fp`

`77 \fp_gset_eq:NN \g__hobby_in_angle_fp \c_inf_fp`

(End of definition for \g__hobby_in_angle_fp.)

`\g__hobby_out_angle_fp` `\g__hobby_out_angle_fp` is the angle at the start of an open path. If this is not specified, it will be computed automatically. It is set to `\c_inf_fp` to allow easy detection of when it has been specified.

`78 \fp_new:N \g__hobby_out_angle_fp`

`79 \fp_gset_eq:NN \g__hobby_out_angle_fp \c_inf_fp`

(End of definition for \g__hobby_out_angle_fp.)

`\g__hobby_npoints_int` `\g__hobby_npoints_int` is one less than the number of points on the curve. As our list of points starts at 0, this is the index of the last point. In the algorithm for a closed curve, some points are repeated whereupon this is incremented so that it is always the index of the last point.

`80 \int_new:N \g__hobby_npoints_int`

(End of definition for \g__hobby_npoints_int.)

\g__hobby_draw_int

81 \int_new:N \g__hobby_draw_int

(End of definition for \g__hobby_draw_int.)

A “point” is a key-value list setting the x-value, the y-value, and the tensions at that point. Using keys makes it easier to pass points from the algorithm code to the calling code and vice versa without either knowing too much about the other.

```

82 \keys_define:nn {hobby / read in all} {
83   point .code:n = {
84     \fp_set:Nn \l_hobby_tempa_fp {\clist_item:nn {#1} {1}}
85     \fp_set:Nn \l_hobby_tempb_fp {\clist_item:nn {#1} {2}}
86   },
87   tension~out .fp_set:N = \l_hobby_tempc_fp,
88   tension~in .fp_set:N = \l_hobby_tempd_fp,
89   excess~angle .fp_set:N = \l_hobby_temps_fp,
90   break .tl_set:N = \l_tmpb_tl,
91   blank .tl_set:N = \l_tmpa_tl,
92   tension .meta:n = { tension~out=#1, tension~in=#1 },
93   break .default:n = false,
94   blank .default:n = false,
95   invert~soft~blanks .choice:,
96   invert~soft~blanks / true .code:n = {
97     \int_gset:Nn \g__hobby_draw_int {0}
98   },
99   invert~soft~blanks / false .code:n = {
100     \int_gset:Nn \g__hobby_draw_int {1}
101   },
102   invert~soft~blanks .default:n = true,
103   tension~out .default:n = 1,
104   tension~in .default:n = 1,
105   excess~angle .default:n = 0,
106   in~angle .fp_gset:N = \g__hobby_in_angle_fp,
107   out~angle .fp_gset:N = \g__hobby_out_angle_fp,
108   in~curl .fp_gset:N = \g__hobby_in_curl_fp,
109   out~curl .fp_gset:N = \g__hobby_out_curl_fp,
110   closed .bool_gset:N = \g__hobby_closed_bool,
111   closed .default:n = true,
112   disjoint .bool_gset:N = \g__hobby_disjoint_bool,
113   disjoint .default:n = true,
114   break~default .code:n = {
115     \keys_define:nn { hobby / read in all }
116     {
117       break .default:n = #1
118     }
119   },
120   blank~default .code:n = {
121     \keys_define:nn { hobby / read in all }
122     {
123       blank .default:n = #1
124     }
125   },
126 }

```

There are certain other parameters that can be set for a given curve.

```

127 \keys_define:nn { hobby / read in params} {
128   in~angle .fp_gset:N = \g__hobby_in_angle_fp,
129   out~angle .fp_gset:N = \g__hobby_out_angle_fp,
130   in~curl .fp_gset:N = \g__hobby_in_curl_fp,
131   out~curl .fp_gset:N = \g__hobby_out_curl_fp,
132   closed .bool_gset:N = \g__hobby_closed_bool,
133   closed .default:n = true,

```

```

134 disjoint .bool_gset:N = \g__hobby_disjoint_bool,
135 disjoint .default:n = true,
136 break-default .code:n = {
137   \keys_define:nn { hobby / read in all }
138   {
139     break .default:n = #1
140   }
141 },
142 blank-default .code:n = {
143   \keys_define:nn { hobby / read in all }
144   {
145     blank .default:n = #1
146   }
147 },
148 invert-soft-blanks .choice:,
149 invert-soft-blanks / true .code:n = {
150   \int_gset:Nn \g__hobby_draw_int {0}
151 },
152 invert-soft-blanks / false .code:n = {
153   \int_gset:Nn \g__hobby_draw_int {1}
154 },
155 invert-soft-blanks .default:n = true,
156 }

```

\hobby_distangle:n Computes the distance and angle between successive points. The argument given is the index of the current point. Assumptions: the points are in `\g__hobby_points_x_array` and `\g__hobby_points_y_array` and the index of the last point is `\g__hobby_npoints_int`.

```

157 \cs_set:Nn \hobby_distangle:n {
158   \fp_set:Nn \l_hobby_tempa_fp {
159     (\array_get:Nn \g__hobby_points_x_array {#1 + 1})
160     - (\array_get:Nn \g__hobby_points_x_array {#1})}
161
162   \fp_set:Nn \l_hobby_tempb_fp {
163     (\array_get:Nn \g__hobby_points_y_array {#1 + 1})
164     - (\array_get:Nn \g__hobby_points_y_array {#1})}
165
166   \fp_set:Nn \l_hobby_tempc_fp { atan ( \l_hobby_tempb_fp, \l_hobby_tempa_fp ) }
167   \fp_veclen:NVV \l_hobby_tempd_fp \l_hobby_tempa_fp \l_hobby_tempb_fp
168
169   \array_gpush:Nx \g__hobby_angles_array {\fp_to_tl:N \l_hobby_tempc_fp}
170   \array_gpush:Nx \g__hobby_distances_array {\fp_to_tl:N \l_hobby_tempd_fp}
171 }

```

(End of definition for \hobby_distangle:n. This function is documented on page ??.)

\fp_veclen:NVV Computes the length of the vector specified by the latter two arguments, storing the answer in the first.

```

172 \cs_new:Nn \fp_veclen:Nnn {
173   \fp_set:Nn #1 {((#2)^2 + (#3)^2)^.5}
174 }
175 \cs_generate_variant:Nn \fp_veclen:Nnn {NVV}

```

(End of definition for \fp_veclen:NVV. This function is documented on page ??.)

\hobby_ctrlen:Nnn Computes the length of the control point vector from the two angles, storing the answer in the first argument given.

```

176 \cs_new:Nn \hobby_ctrlen:Nnn {
177   \fp_set:Nn #1 {(2 - \g_hobby_parama_fp
178     * ( sin(#2) - \g_hobby_paramb_fp * sin(#3) )
179     * ( sin(#3) - \g_hobby_paramb_fp * sin(#2) )
180     * ( cos(#2) - cos(#3) ) ) }

```

```

181      / ( 1 + (1 - \g_hobby_paramc_fp) * cos(#3) + \g_hobby_paramc_fp * cos(#2))}
182    }
183    \cs_generate_variant:Nn \hobby_ctrllen:Nnn {NVV}

```

(End of definition for \hobby_ctrllen:Nnn. This function is documented on page ??.)

by_append_point_copy:n This function adds a copy of the point (numbered by its argument) to the end of the list of points, copying all the relevant data (coordinates, tension, etc.).

Originally from Bruno Le Foch on TeX-SX.

```

184 \cs_new_protected:Npn \hobby_append_point_copy:n #1
185 {
186   \hobby_append_point_copy_aux:Nn \g__hobby_points_array {#1}
187   \hobby_append_point_copy_aux:Nn \g__hobby_points_x_array {#1}
188   \hobby_append_point_copy_aux:Nn \g__hobby_points_y_array {#1}
189   \hobby_append_point_copy_aux:Nn \g__hobby_tension_in_array {#1}
190   \hobby_append_point_copy_aux:Nn \g__hobby_tension_out_array {#1}
191   \hobby_append_point_copy_aux:Nn \g__hobby_excess_angle_array {#1}
192   \hobby_append_point_copy_aux:Nn \g__hobby_actions_array {#1}
193 }
194 \cs_new_protected:Npn \hobby_append_point_copy_aux:Nn #1#2
195 { \array_gpush:Nx #1 { \array_get:Nn #1 {#2} } }

```

(End of definition for \hobby_append_point_copy:n. This function is documented on page ??.)

\hobby_gen_path: This is the curve generation function. We assume at the start that we have an array containing all the points that the curve must go through, and the various curve parameters have been initialised. So these must be set up by a wrapper function which then calls this one. The list of required information is:

1. \g__hobby_points_x_array
2. \g__hobby_points_y_array
3. \g__hobby_tension_out_array
4. \g__hobby_tension_in_array
5. \g__hobby_excess_angle_array
6. \g__hobby_in_curl_fp
7. \g__hobby_out_curl_fp
8. \g__hobby_in_angle_fp
9. \g__hobby_out_angle_fp
10. \g__hobby_closed_bool
11. \g__hobby_actions_array

```

196 \cs_new:Nn \hobby_gen_path:
197 {

```

For much of the time, we can pretend that a closed path is the same as an open path. To do this, we need to make the end node an internal node by repeating the z_1 node as the z_{n+1} th node. We also check that the last (z_n) and first (z_0) nodes are the same, otherwise we repeat the z_0 node as well.

```

198 \bool_if:NT \g__hobby_closed_bool {

```

Are the x -values of the first and last points different?

```

199   \fp_compare:nTF {(\array_get:Nn \g__hobby_points_x_array {0})}
200   =
201   {(\array_top:N \g__hobby_points_x_array)}
202   {

```


No, so compare the y -values. Are the y -values of the first and last points different?

```

203     \fp_compare:nF {
204         \array_get:Nn \g__hobby_points_y_array {0}
205         =
206         \array_top:N \g__hobby_points_y_array
207     }
208     {

```

Yes, so we need to duplicate the first point, with all of its data.

```

209         \hobby_append_point_copy:n {0}
210     }
211 }
212 {

```

Yes, so we need to duplicate the first point, with all of its data.

```

213         \hobby_append_point_copy:n {0}
214     }

```

Now that we are sure that the first and last points are identical, we need to duplicate the first-but-one point (and all of its data).

```

215         \hobby_append_point_copy:n {1}
216     }

```

Set `\g__hobby_npoints_int` to the number of points (minus one).

```

217 \int_gset:Nn \g__hobby_npoints_int {\array_length:N \g__hobby_points_y_array}

```

At this point, we need to decide what to do. This will depend on whether we have any intermediate points.

```

218 \int_compare:nNnTF {\g__hobby_npoints_int} = {0} {

```

Only one point, do nothing

```

219 }
220 {
221     \int_compare:nNnTF {\g__hobby_npoints_int} = {1} {

```

Only two points, skip processing. Just need to set the incoming and outgoing angles

```

222 \hobby_distangle:n {0}
223 \fp_compare:nF { \g__hobby_out_angle_fp == \c_inf_fp }
224 {
225     \fp_set:Nn \l_hobby_tempa_fp { \g__hobby_out_angle_fp
226         - \array_get:Nn \g__hobby_angles_array {0}}

```

We want to ensure that these angles lie in the range $(-\pi, \pi]$. So if the angle is bigger than π , we subtract 2π . (It shouldn't be that we can get bigger than 3π - check this)

```

227     \fp_compare:nT {\l_hobby_tempa_fp > \c_pi_fp }
228     {
229         \fp_sub:Nn \l_hobby_tempa_fp {2 * \c_pi_fp}
230     }

```

Similarly, we check to see if the angle is less than $-\pi$.

```

231     \fp_compare:nT {\l_hobby_tempa_fp < -\c_pi_fp }
232     {
233         \fp_add:Nn \l_hobby_tempa_fp {2 * \c_pi_fp}
234     }
235     \array_gput:Nnx \g__hobby_theta_array {0} {\fp_to_tl:N \l_hobby_tempa_fp}
236     \fp_compare:nT { \g__hobby_in_angle_fp == \c_inf_fp }
237     {
238         %%A \fp_mul:Nn \l_hobby_tempa_fp {-1}
239         \array_gput:Nnx \g__hobby_phi_array {1}{ \fp_to_tl:N \l_hobby_tempa_fp}
240     }
241 }
242 \fp_compare:nTF { \g__hobby_in_angle_fp == \c_inf_fp }
243 {
244     \fp_compare:nT { \g__hobby_out_angle_fp == \c_inf_fp }

```

```

245 {
246   \array_gput:Nnx \g__hobby_phi_array {1} {0}
247   \array_gput:Nnx \g__hobby_theta_array {0} {0}
248 }
249 }
250 {
251   \fp_set:Nn \l_hobby_tempa_fp { - \g__hobby_in_angle_fp + \c_pi_fp
252 + (\array_get:Nn \g__hobby_angles_array {0})}
253   \fp_compare:nT {\l_hobby_tempa_fp > \c_pi_fp }
254   {
255     \fp_sub:Nn \l_hobby_tempa_fp {2 * \c_pi_fp}
256   }
257   \fp_compare:nT {\l_hobby_tempa_fp < -\c_pi_fp }
258   {
259     \fp_add:Nn \l_hobby_tempa_fp {2 * \c_pi_fp}
260   }
261
262   \array_gput:Nnx \g__hobby_phi_array {1}
263   {\fp_to_tl:N \l_hobby_tempa_fp}
264   \fp_compare:nT { \g__hobby_out_angle_fp == \c_inf_fp }
265   {
266     %%~A      \fp_mul:Nn \l_hobby_tempa_fp {-1}
267     \array_gput:Nnx \g__hobby_theta_array {0}{\fp_to_tl:N \l_hobby_tempa_fp}
268   }
269 }
270
271 }
272 {

```

Got enough points, go on with processing

```

273   \hobby_compute_path:
274 }
275 \hobby_build_path:
276 }
277 }

```

(End of definition for \hobby_gen_path:. This function is documented on page ??.)

\hobby_compute_path: This is the path builder where we have enough points to run the algorithm.

```

278 \cs_new:Nn \hobby_compute_path:
279 {

```

Our first step is to go through the list of points and compute the distances and angles between successive points. Thus d_i is the distance from z_i to z_{i+1} and the angle is the angle of the line from z_i to z_{i+1} .

```

280 \int_step_function:nnnN {0} {1} {\g__hobby_npoints_int - 1} \hobby_distangle:n

```

For the majority of the code, we're only really interested in the differences of the angles. So for each internal point we compute the differences in the angles.

```

281 \int_step_inline:nnnn {1} {1} {\g__hobby_npoints_int - 1} {
282   \fp_set:Nx \l_hobby_tempa_fp {
283     \array_get:Nn \g__hobby_angles_array {##1}
284     - \array_get:Nn \g__hobby_angles_array {##1 - 1}}

```

We want to ensure that these angles lie in the range $(-\pi, \pi]$. So if the angle is bigger than π , we subtract 2π . (It shouldn't be that we can get bigger than 3π - check this.)

```

285   \fp_compare:nTF {\l_hobby_tempa_fp > \c_pi_fp }
286   {
287     \fp_sub:Nn \l_hobby_tempa_fp {2 * \c_pi_fp}
288   }
289   {}

```

Similarly, we check to see if the angle is less than $-\pi$.

```

290     \fp_compare:nTF {\l_hobby_tempa_fp <= -\c_pi_fp }
291     {
292         \fp_add:Nn \l_hobby_tempa_fp {2 * \c_pi_fp}
293     }
294     {}

```

The wrapping routine might not get it right at the edges so we add in the override.

```

295 \array_get:NnNTF \g__hobby_excess_angle_array {##1} \l_tmpa_tl {
296     \fp_add:Nn \l_hobby_tempa_fp {2 * \c_pi_fp * \l_tmpa_tl}
297 }{}
298 \array_gput:Nnx \g__hobby_psi_array {##1}{\fp_to_tl:N \l_hobby_tempa_fp}
299 }

```

Next, we generate the matrix. We start with the subdiagonal. This is indexed from 1 to $n - 1$.

```

300 \int_step_inline:nnnn {1} {1} {\g__hobby_npoints_int - 1} {
301     \array_gput:Nnx \g__hobby_matrix_a_array {##1} {\fp_to_tl:n {
302         \array_get:Nn \g__hobby_tension_in_array {##1}^2
303         * \array_get:Nn \g__hobby_distances_array {##1}
304         * \array_get:Nn \g__hobby_tension_in_array {##1 + 1}
305     }}
306 }

```

Next, we attack main diagonal. We might need to adjust the first and last terms, but we'll do that in a minute.

```

307 \int_step_inline:nnnn {1} {1} {\g__hobby_npoints_int - 1} {
308
309     \array_gput:Nnx \g__hobby_matrix_b_array {##1} {\fp_to_tl:n
310     {(3 * (\array_get:Nn \g__hobby_tension_in_array {##1 + 1}) - 1) *
311     (\array_get:Nn \g__hobby_tension_out_array {##1})^2 *
312     (\array_get:Nn \g__hobby_tension_out_array {##1 - 1})
313     * ( \array_get:Nn \g__hobby_distances_array {##1 - 1})
314     +
315     (3 * (\array_get:Nn \g__hobby_tension_out_array {##1 - 1}) - 1)
316     * (\array_get:Nn \g__hobby_tension_in_array {##1})^2
317     * (\array_get:Nn \g__hobby_tension_in_array {##1 + 1})
318     * (\array_get:Nn \g__hobby_distances_array {##1})}}
319 }
320 }

```

Next, the superdiagonal.

```

321 \int_step_inline:nnnn {1} {1} {\g__hobby_npoints_int - 2} {
322
323     \array_gput:Nnx \g__hobby_matrix_c_array {##1} {\fp_to_tl:n
324     {(\array_get:Nn \g__hobby_tension_in_array {##1})^2
325     * (\array_get:Nn \g__hobby_tension_in_array {##1 - 1})
326     * (\array_get:Nn \g__hobby_distances_array {##1 - 1})
327     }}
328
329 }

```

Lastly (before the adjustments), the target vector.

```

330 \int_step_inline:nnnn {1} {1} {\g__hobby_npoints_int - 2} {
331
332     \array_gput:Nnx \g__hobby_matrix_d_array {##1} {\fp_to_tl:n
333     {
334     - (\array_get:Nn \g__hobby_psi_array {##1 + 1})
335     * (\array_get:Nn \g__hobby_tension_out_array {##1})^2
336     * (\array_get:Nn \g__hobby_tension_out_array {##1 - 1})
337     * (\array_get:Nn \g__hobby_distances_array {##1 - 1})
338     - (3 * (\array_get:Nn \g__hobby_tension_out_array {##1 - 1}) - 1)
339     * (\array_get:Nn \g__hobby_psi_array {##1})

```

```

340 * (\array_get:Nn \g__hobby_tension_in_array {##1})^2
341 * (\array_get:Nn \g__hobby_tension_in_array {##1 + 1})
342 * (\array_get:Nn \g__hobby_distances_array {##1})
343 }
344 }
345 }

```

Next, there are some adjustments at the ends. These differ depending on whether the path is open or closed.

```

346 \bool_if:NTF \g__hobby_closed_bool {
Closed path
347 \array_gput:Nnx \g__hobby_matrix_c_array {0} {\fp_to_tl:n {
348 - (\array_get:Nn \g__hobby_distances_array {\g__hobby_npoints_int - 2})
349 * (\array_get:Nn \g__hobby_tension_out_array {\g__hobby_npoints_int - 2})
350 * (\array_get:Nn \g__hobby_tension_out_array {\g__hobby_npoints_int - 1})^2
351 }}
352
353 \array_gput:Nnn \g__hobby_matrix_b_array {0} {1}
354 \array_gput:Nnn \g__hobby_matrix_d_array {0} {0}
355
356 \array_gput:Nnx \g__hobby_matrix_b_array {\g__hobby_npoints_int - 1} {\fp_to_tl:n {
357 (\array_get:Nn \g__hobby_matrix_b_array {\g__hobby_npoints_int - 1})
358 + 1
359 }}
360
361 \array_gput:Nnx \g__hobby_matrix_d_array {\g__hobby_npoints_int - 1} {\fp_to_tl:n {
362 - (\array_get:Nn \g__hobby_psi_array {1})
363 * (\array_get:Nn \g__hobby_tension_out_array {\g__hobby_npoints_int - 1})^2
364 * (\array_get:Nn \g__hobby_tension_out_array {\g__hobby_npoints_int - 2})
365 * (\array_get:Nn \g__hobby_distances_array {\g__hobby_npoints_int - 2})
366 - (3 * (\array_get:Nn \g__hobby_tension_out_array {\g__hobby_npoints_int - 2}) - 1)
367 * (\array_get:Nn \g__hobby_psi_array {\g__hobby_npoints_int - 1})
368 * (\array_get:Nn \g__hobby_tension_in_array {\g__hobby_npoints_int - 1})^2
369 * (\array_get:Nn \g__hobby_tension_in_array {\g__hobby_npoints_int})
370 * (\array_get:Nn \g__hobby_distances_array {\g__hobby_npoints_int - 1})
371 }
372 }

```

We also need to populate the u -vector

```

373 \array_gput:Nnn \g__hobby_vector_u_array {0} {1}
374 \array_gput:Nnn \g__hobby_vector_u_array {\g__hobby_npoints_int - 1} {1}
375 \int_step_inline:nnnn {1} {1} {\g__hobby_npoints_int - 2} {
376 \array_gput:Nnn \g__hobby_vector_u_array {##1} {0}
377 }

```

And define the significant entry in the v -vector.

```

378 \fp_set:Nn \l_hobby_matrix_v_fp {
379 (\array_get:Nn \g__hobby_tension_out_array {\g__hobby_npoints_int - 1})^2
380 * (\array_get:Nn \g__hobby_tension_out_array {\g__hobby_npoints_int - 2})
381 * (\array_get:Nn \g__hobby_distances_array {\g__hobby_npoints_int - 2})
382 }
383 }
384 {

```

Open path. First, we test to see if θ_0 has been specified.

```

385 \fp_compare:nTF { \g__hobby_out_angle_fp == \c_inf_fp }
386 {
387 \array_gput:Nnx \g__hobby_matrix_b_array {0} {\fp_to_tl:n {
388 (\array_get:Nn \g__hobby_tension_in_array {1})^3
389 * \g__hobby_in_curl_fp
390 +
391 (3 * (\array_get:Nn \g__hobby_tension_in_array {1}) - 1)

```

```

392 * (\array_get:Nn \g__hobby_tension_out_array {0})^3
393 }}
394
395 \array_gput:Nnx \g__hobby_matrix_c_array {0} {\fp_to_tl:n {
396 (\array_get:Nn \g__hobby_tension_out_array {0})^3
397 +
398 (3 * (\array_get:Nn \g__hobby_tension_out_array {0}) - 1)
399 * (\array_get:Nn \g__hobby_tension_in_array {1})^3
400 * \g__hobby_in_curl_fp
401 }}
402
403 \array_gput:Nnx \g__hobby_matrix_d_array {0} {\fp_to_tl:n {
404 -( (\array_get:Nn \g__hobby_tension_out_array {0})^3
405 +
406 (3 * (\array_get:Nn \g__hobby_tension_out_array {0}) - 1)
407 * (\array_get:Nn \g__hobby_tension_in_array {1})^3
408 * \g__hobby_in_curl_fp)
409 * (\array_get:Nn \g__hobby_psi_array {1})
410 }}
411
412 }
413 {
414 \array_gput:Nnn \g__hobby_matrix_b_array {0} {1}
415 \array_gput:Nnn \g__hobby_matrix_c_array {0} {0}
416 \fp_set:Nn \l_hobby_tempa_fp { \g__hobby_out_angle_fp
417 - \array_get:Nn \g__hobby_angles_array {0}}

```

We want to ensure that these angles lie in the range $(-\pi, \pi]$. So if the angle is bigger than π , we subtract 2π . (It shouldn't be that we can get bigger than 3π - check this)

```

418 \fp_compare:nT {\l_hobby_tempa_fp > \c_pi_fp }
419 {
420 \fp_sub:Nn \l_hobby_tempa_fp {2 * \c_pi_fp}
421 }

```

Similarly, we check to see if the angle is less than $-\pi$.

```

422 \fp_compare:nT {\l_hobby_tempa_fp < -\c_pi_fp }
423 {
424 \fp_add:Nn \l_hobby_tempa_fp {2 * \c_pi_fp}
425 }
426 \array_gput:Nnx \g__hobby_matrix_d_array {0} {\fp_to_tl:N \l_hobby_tempa_fp}
427 }

```

Next, if ϕ_n has been given.

```

428 \fp_compare:nTF { \g__hobby_in_angle_fp == \c_inf_fp }
429 {
430
431 \array_gput:Nnx \g__hobby_matrix_b_array {\g__hobby_npoints_int - 1} {\fp_to_tl:n {
432 \array_get:Nn \g__hobby_matrix_b_array {\g__hobby_npoints_int - 1}
433 - (\array_get:Nn \g__hobby_tension_out_array {\g__hobby_npoints_int - 1})^2
434 * (\array_get:Nn \g__hobby_tension_out_array {\g__hobby_npoints_int - 2})
435 * (\array_get:Nn \g__hobby_distances_array {\g__hobby_npoints_int - 2})
436 *
437 ((3 * (\array_get:Nn \g__hobby_tension_in_array {\g__hobby_npoints_int} ) - 1)
438 * (\array_get:Nn \g__hobby_tension_out_array {\g__hobby_npoints_int - 1})^3 \l_tmpa_tl
439 * \g__hobby_out_curl_fp
440 +
441 (\array_get:Nn \g__hobby_tension_in_array {\g__hobby_npoints_int} )^3)
442 /
443 ((3 * (\array_get:Nn \g__hobby_tension_out_array {\g__hobby_npoints_int - 2}) - 1)
444 * (\array_get:Nn \g__hobby_tension_in_array {\g__hobby_npoints_int})^3
445 +
446 ( \array_get:Nn \g__hobby_tension_out_array {\g__hobby_npoints_int - 1})^3

```

```

447 * \g__hobby_out_curl_fp)
448 }}
449
450 \array_gput:Nnx \g__hobby_matrix_d_array {\g__hobby_npoints_int - 1} {\fp_to_tl:n {
451 - (3 * (\array_get:Nn \g__hobby_tension_out_array {\g__hobby_npoints_int - 2}) - 1)
452 * (\array_get:Nn \g__hobby_psi_array {\g__hobby_npoints_int - 1})
453 * (\array_get:Nn \g__hobby_tension_in_array {\g__hobby_npoints_int - 1})^2
454 * (\array_get:Nn \g__hobby_tension_in_array {\g__hobby_npoints_int})
455 * (\array_get:Nn \g__hobby_distances_array {\g__hobby_npoints_int - 1})
456 }}
457
458 }
459 {
460 \fp_set:Nn \l_hobby_tempa_fp { - \g__hobby_in_angle_fp + \c_pi_fp
461 + (\array_get:Nn \g__hobby_angles_array {\g__hobby_npoints_int - 1})}
462 \fp_compare:nT {\l_hobby_tempa_fp > \c_pi_fp }
463 {
464 \fp_sub:Nn \l_hobby_tempa_fp {2 * \c_pi_fp}
465 }
466 \fp_compare:nT {\l_hobby_tempa_fp < -\c_pi_fp }
467 {
468 \fp_add:Nn \l_hobby_tempa_fp {2 * \c_pi_fp}
469 }
470
471 \array_gput:Nnx \g__hobby_phi_array {\g__hobby_npoints_int}
472 {\fp_to_tl:N \l_hobby_tempa_fp}
473
474 \array_gput:Nnx \g__hobby_matrix_d_array {\g__hobby_npoints_int - 1} {\fp_to_tl:n {
475 \l_hobby_tempa_fp
476 * (\array_get:Nn \g__hobby_tension_out_array {\g__hobby_npoints_int - 1})^2
477 * (\array_get:Nn \g__hobby_tension_out_array {\g__hobby_npoints_int - 2})
478 * (\array_get:Nn \g__hobby_distances_array {\g__hobby_npoints_int - 2})
479 -
480 (3 * ( \array_get:Nn \g__hobby_tension_out_array {\g__hobby_npoints_int - 2}) - 1)
481 * (\array_get:Nn \g__hobby_psi_array {\g__hobby_npoints_int - 1})
482 * (\array_get:Nn \g__hobby_tension_in_array {\g__hobby_npoints_int - 1})^2
483 * (\array_get:Nn \g__hobby_tension_in_array {\g__hobby_npoints_int})
484 * (\array_get:Nn \g__hobby_distances_array {\g__hobby_npoints_int - 1}) }}
485 }

```

End of adjustments for open paths.

```

486 }

```

Now we have the tridiagonal matrix in place, we implement the solution. We start with the forward eliminations.

```

487 \int_step_inline:nnnn {1} {1} {\g__hobby_npoints_int - 1} {
488
489 \array_gput:Nnx \g__hobby_matrix_b_array {##1} {\fp_to_tl:n {
490 (\array_get:Nn \g__hobby_matrix_b_array {##1 - 1})
491 * (\array_get:Nn \g__hobby_matrix_b_array {##1})
492 -
493 (\array_get:Nn \g__hobby_matrix_c_array {##1 - 1})
494 * (\array_get:Nn \g__hobby_matrix_a_array {##1})
495 }}

```

The last time, we don't touch the C -vector.

```

496 \int_compare:nT {##1 < \g__hobby_npoints_int - 1} {
497
498 \array_gput:Nnx \g__hobby_matrix_c_array {##1} {\fp_to_tl:n {
499 (\array_get:Nn \g__hobby_matrix_b_array {##1 - 1})
500 * (\array_get:Nn \g__hobby_matrix_c_array {##1})
501 }}

```

```

502 }
503
504 \array_gput:Nnx \g__hobby_matrix_d_array {##1} {\fp_to_tl:n {
505 (\array_get:Nn \g__hobby_matrix_b_array {##1 - 1})
506 * (\array_get:Nn \g__hobby_matrix_d_array {##1})
507 -
508 (\array_get:Nn \g__hobby_matrix_d_array {##1 - 1})
509 * (\array_get:Nn \g__hobby_matrix_a_array {##1})
510 }}

```

On a closed path, we also want to know $M^{-1}u$ so need to do the elimination steps on u as well.

```

511 \bool_if:NT \g__hobby_closed_bool {
512 \array_gput:Nnx \g__hobby_vector_u_array {##1} {\fp_to_tl:n {
513 (\array_get:Nn \g__hobby_matrix_b_array {##1 - 1})
514 * (\array_get:Nn \g__hobby_vector_u_array {##1})
515 -
516 (\array_get:Nn \g__hobby_vector_u_array {##1 - 1})
517 * (\array_get:Nn \g__hobby_matrix_a_array {##1})
518 }}
519 }
520 }

```

Now we start the back substitution. The first step is slightly different to the general step.

```

521 \array_gput:Nnx \g__hobby_theta_array {\g__hobby_npoints_int - 1} {\fp_to_tl:n {
522 (\array_get:Nn \g__hobby_matrix_d_array {\g__hobby_npoints_int - 1})
523 / (\array_get:Nn \g__hobby_matrix_b_array {\g__hobby_npoints_int - 1})
524 }}

```

For a closed path, we need to work with u as well.

```

525 \bool_if:NT \g__hobby_closed_bool {
526 \array_gput:Nnx \g__hobby_vector_u_array {\g__hobby_npoints_int - 1} {\fp_to_tl:n {
527 (\array_get:Nn \g__hobby_vector_u_array {\g__hobby_npoints_int - 1})
528 / (\array_get:Nn \g__hobby_matrix_b_array {\g__hobby_npoints_int - 1})
529 }}
530 }

```

Now we iterate over the vectors, doing the remaining back substitutions.

```

531 \int_step_inline:nnnn {\g__hobby_npoints_int - 2} {-1} {0} {
532
533 \array_gput:Nnx \g__hobby_theta_array {##1} {\fp_to_tl:n {
534 ( (\array_get:Nn \g__hobby_matrix_d_array {##1})
535 - (\array_get:Nn \g__hobby_theta_array {##1 + 1})
536 * (\array_get:Nn \g__hobby_matrix_c_array {##1})
537 ) / (\array_get:Nn \g__hobby_matrix_b_array {##1})
538 }}
539 }
540 \bool_if:NT \g__hobby_closed_bool {

```

On a closed path, we also need to work out $M^{-1}u$.

```

541 \int_step_inline:nnnn {\g__hobby_npoints_int - 2} {-1} {0} {
542 \array_gput:Nnx \g__hobby_vector_u_array {##1} {\fp_to_tl:n {
543 {
544 ((\array_get:Nn \g__hobby_vector_u_array {##1})
545 - (\array_get:Nn \g__hobby_vector_u_array {##1 + 1})
546 * (\array_get:Nn \g__hobby_matrix_c_array {##1})
547 ) / (\array_get:Nn \g__hobby_matrix_b_array {##1})
548 }}
549 }

```

Then we compute $v^\top M^{-1}u$ and $v^\top M^{-1}\theta$. As v has a particularly simple form, these inner products are easy to compute.

```

550
551 \fp_set:Nn \l_hobby_tempb_fp {

```

```

552 ((\array_get:Nn \g__hobby_theta_array {1})
553 * \l_hobby_matrix_v_fp
554 - (\array_get:Nn \g__hobby_theta_array {\g__hobby_npoints_int - 1})
555 ) / (
556 (\array_get:Nn \g__hobby_vector_u_array {1})
557 * \l_hobby_matrix_v_fp
558 - (\array_get:Nn \g__hobby_vector_u_array {\g__hobby_npoints_int - 1})
559 + 1
560 )}
561
562 \int_step_inline:nnnn {0} {1} {\g__hobby_npoints_int - 1} {
563
564   \array_gput:Nnx \g__hobby_theta_array {##1} {\fp_to_tl:n {
565     (\array_get:Nn \g__hobby_theta_array {##1})
566     - (\array_get:Nn \g__hobby_vector_u_array {##1})
567     * \l_hobby_tempb_fp
568   }}
569 }
570 }

```

Now that we have computed the θ_i s, we can quickly compute the ϕ_i s.

```

571 \int_step_inline:nnnn {1} {1} {\g__hobby_npoints_int - 1} {
572
573   \array_gput:Nnx \g__hobby_phi_array {##1} {\fp_to_tl:n {
574     - (\array_get:Nn \g__hobby_psi_array {##1})
575     - (\array_get:Nn \g__hobby_theta_array {##1})
576   }}
577 }

```

If the path is open, this works for all except ϕ_n . If the path is closed, we can drop our added point. Cheaply, of course.

```

578 \bool_if:NTF \g__hobby_closed_bool {
579   \int_gdecr:N \g__hobby_npoints_int
580 }{

```

If ϕ_n was not given, we compute it from θ_{n-1} .

```

581 \fp_compare:nT { \g__hobby_in_angle_fp == \c_inf_fp }
582 {
583   \array_gput:Nnx \g__hobby_phi_array {\g__hobby_npoints_int} {\fp_to_tl:n {
584     ((3 * (\array_get:Nn \g__hobby_tension_in_array {\g__hobby_npoints_int}) - 1)
585     * (\array_get:Nn \g__hobby_tension_out_array {\g__hobby_npoints_int - 1})^3
586     * \g__hobby_out_curl_fp
587     +
588     (\array_get:Nn \g__hobby_tension_in_array {\g__hobby_npoints_int})^3)
589     /
590     ((3 * (\array_get:Nn \g__hobby_tension_out_array {\g__hobby_npoints_int - 2}) - 1)
591     * (\array_get:Nn \g__hobby_tension_in_array {\g__hobby_npoints_int})^3 \l_tmpa_tl
592     +
593     (\array_get:Nn \g__hobby_tension_out_array {\g__hobby_npoints_int - 1})^3
594     * \g__hobby_out_curl_fp)
595     *
596     (\array_get:Nn \g__hobby_theta_array {\g__hobby_npoints_int - 1})
597   }}
598 }
599 }
600 }

```

(End of definition for `\hobby_compute_path`:. This function is documented on page ??.)

`\hobby_build_path`: Once we've computed the angles, we build the actual path.

```

601 \cs_new:Nn \hobby_build_path:
602 {

```


Next task is to compute the ρ_i and σ_i .

```

603 \int_step_inline:nnnn {0} {1} {\g__hobby_npoints_int - 1} {
604
605   \fp_set:Nn \l_hobby_tempa_fp {\array_get:Nn \g__hobby_theta_array {##1}}
606
607   \fp_set:Nn \l_hobby_tempb_fp {\array_get:Nn \g__hobby_phi_array {##1 + 1}}
608
609   \hobby_ctrlilen:NVV \l_hobby_temps_fp \l_hobby_tempa_fp \l_hobby_tempb_fp
610
611   \array_gput:Nnx \g__hobby_sigma_array {##1 + 1} {\fp_to_tl:N \l_hobby_temps_fp}
612
613   \hobby_ctrlilen:NVV \l_hobby_temps_fp \l_hobby_tempb_fp \l_hobby_tempa_fp
614
615   \array_gput:Nnx \g__hobby_rho_array {##1} {\fp_to_tl:N \l_hobby_temps_fp}
616
617 }

```

Lastly, we generate the coordinates of the control points.

```

618 \int_step_inline:nnnn {0} {1} {\g__hobby_npoints_int - 1} {
619   \array_gput:Nnx \g__hobby_controla_array {##1 + 1} {\fp_eval:n {
620     (\array_get:Nn \g__hobby_points_x_array {##1})
621     +
622     (\array_get:Nn \g__hobby_distances_array {##1}) *
623     (\array_get:Nn \g__hobby_rho_array {##1}) *
624     cos ( (\array_get:Nn \g__hobby_angles_array {##1})
625     +
626     (\array_get:Nn \g__hobby_theta_array {##1}))
627   /3
628 }, \fp_eval:n {
629   (\array_get:Nn \g__hobby_points_y_array {##1}) +
630   (\array_get:Nn \g__hobby_distances_array {##1}) *
631   (\array_get:Nn \g__hobby_rho_array {##1}) *
632   sin ( (\array_get:Nn \g__hobby_angles_array {##1})
633   +
634   (\array_get:Nn \g__hobby_theta_array {##1}))
635 /3
636 }
637 }
638 }
639 \int_step_inline:nnnn {1} {1} {\g__hobby_npoints_int} {
640   \array_gput:Nnx \g__hobby_controlb_array {##1} {
641     \fp_eval:n {\array_get:Nn \g__hobby_points_x_array {##1}
642       - (\array_get:Nn \g__hobby_distances_array {##1 - 1})
643       * (\array_get:Nn \g__hobby_sigma_array {##1})
644       * cos((\array_get:Nn \g__hobby_angles_array {##1 - 1})
645       - (\array_get:Nn \g__hobby_phi_array {##1}))/3
646     }, \fp_eval:n {
647       (\array_get:Nn \g__hobby_points_y_array {##1})
648       - (\array_get:Nn \g__hobby_distances_array {##1 - 1})
649       * (\array_get:Nn \g__hobby_sigma_array {##1})
650       * sin((\array_get:Nn \g__hobby_angles_array {##1 - 1})
651       - (\array_get:Nn \g__hobby_phi_array {##1}))/3
652     } }
653 }
654 }

```

(End of definition for \hobby_build_path:. This function is documented on page ??.)

\hobbyinit Initialise the settings for Hobby's algorithm

```

655 \NewDocumentCommand \hobbyinit {m m m} {
656   \hobby_set_cmds:NNN #1#2#3

```

```

657 \hobby_clear_path:
658 }

```

(End of definition for \hobbyinit. This function is documented on page ??.)

\hobbyaddpoint This adds a point, possibly with tensions, to the current stack.

```

659 \NewDocumentCommand \hobbyaddpoint { m } {
660   \keys_set:nn { hobby/read in all }
661   {
662     tension~out,
663     tension~in,
664     excess~angle,
665     blank,
666     break,
667     #1
668   }
669   \tl_if_eq:VnTF \l_tmpa_tl {true}
670   {\tl_set:Nn \l_tmpa_tl {2}}
671   {
672     \tl_if_eq:VnTF \l_tmpa_tl {soft}
673     {\tl_set:Nn \l_tmpa_tl {0}}
674     {\tl_set:Nn \l_tmpa_tl {1}}
675   }
676   \tl_if_eq:VnTF \l_tmpb_tl {true}
677   {\tl_put_right:Nn \l_tmpa_tl {1}}
678   {\tl_put_right:Nn \l_tmpa_tl {0}}
679   \tl_set:Nx \l_hobby_tempa_tl {\fp_use:N \l_hobby_tempa_fp}
680   \tl_set:Nx \l_hobby_tempb_tl {\fp_use:N \l_hobby_tempb_fp}
681   \hobby_add_point:VVVVV \l_hobby_tempa_tl \l_hobby_tempb_tl \l_hobby_tempc_fp \l_hobby_tempd_fp
682 }

```

(End of definition for \hobbyaddpoint. This function is documented on page ??.)

\hobby_add_point:n

```

683 \cs_new_nopar:Npn \hobby_add_point:nnnnnn #1#2#3#4#5#6
684 {
685   \array_gpush:Nn \g__hobby_actions_array { #6 }
686   \array_gpush:Nn \g__hobby_tension_out_array { #3 }
687   \array_gpush:Nn \g__hobby_tension_in_array { #4 }
688   \array_gpush:Nn \g__hobby_excess_angle_array { #5 }
689   \array_gpush:Nn \g__hobby_points_array { #1, #2 }
690   \array_gpush:Nn \g__hobby_points_x_array { #1 }
691   \array_gpush:Nn \g__hobby_points_y_array { #2 }
692 }
693 \cs_generate_variant:Nn \hobby_add_point:nnnnnn {VVVVVV}

```

(End of definition for \hobby_add_point:n. This function is documented on page ??.)

\hobbysetparams This sets the parameters for the curve.

```

694 \NewDocumentCommand \hobbysetparams { m } {
695   \keys_set:nn { hobby / read in params }
696   {
697     #1
698   }
699 }

```

(End of definition for \hobbysetparams. This function is documented on page ??.)

\hobby_set_cmds:NNN The path-generation code doesn't know what to actually do with the path so the initialisation code will set some macros to do that. This is an auxiliary command that sets these macros.

```

700 \cs_new:Npn \hobby_moveto:nnn #1#2#3 {}
701 \cs_new:Npn \hobby_curveto:nnn #1#2#3 {}

```

```

702 \cs_new:Npn \hobby_close:n #1 {}
703 \cs_generate_variant:Nn \hobby_moveto:nnn {VVV,nnV}
704 \cs_generate_variant:Nn \hobby_curveto:nnn {VVV}
705 \cs_generate_variant:Nn \hobby_close:n {V}
706 \cs_new:Nn \hobby_set_cmds:NNN {
707   \cs_gset_eq:NN \hobby_moveto:nnn #1
708   \cs_gset_eq:NN \hobby_curveto:nnn #2
709   \cs_gset_eq:NN \hobby_close:n #3
710 }

```

(End of definition for \hobby_set_cmds:NNN. This function is documented on page ??.)

\hobbygenpath This is the user (well, sort of) command that generates the curve.

```

711 \NewDocumentCommand \hobbygenpath { } {
712   \array_if_empty:NF \g__hobby_points_array {
713     \hobby_gen_path:
714   }
715 }

```

(End of definition for \hobbygenpath. This function is documented on page ??.)

\hobbygenifnecpath If the named path doesn't exist, it is generated and named. If it does exist, we restore it. Either way, we save it to the aux file.

```

716 \NewDocumentCommand \hobbygenifnecpath { m } {
717   \tl_if_exist:cTF {g_hobby_#1_path}
718   {
719     \tl_use:c {g_hobby_#1_path}
720   }
721   {
722     \hobby_gen_path:
723   }
724   \hobby_save_path:n {#1}
725   \hobby_save_path_to_aux:x {#1}
726 }

```

(End of definition for \hobbygenifnecpath. This function is documented on page ??.)

\hobbygenifnecusepath If the named path doesn't exist, it is generated and named. If it does exist, we restore it. Either way, we save it to the aux file.

```

727 \NewDocumentCommand \hobbygenuseifnecpath { m } {
728   \tl_if_exist:cTF {g_hobby_#1_path}
729   {
730     \tl_use:c {g_hobby_#1_path}
731   }
732   {
733     \hobby_gen_path:
734   }
735   \hobby_save_path:n {#1}
736   \hobby_save_path_to_aux:x {#1}
737   \hobby_use_path:
738 }

```

(End of definition for \hobbygenifnecusepath. This function is documented on page ??.)

\hobbyusepath This is the user (well, sort of) command that uses the last generated curve.

```

739 \NewDocumentCommand \hobbyusepath { m } {
740   \hobbysetparams{#1}
741   \hobby_use_path:
742 }

```

(End of definition for \hobbyusepath. This function is documented on page ??.)

`\hobbysavepath` This is the user (well, sort of) command that uses the last generated curve.

```

743 \NewDocumentCommand \hobbysavepath { m } {
744   \hobby_save_path:n {#1}
745 }

```

(End of definition for \hobbysavepath. This function is documented on page ??.)

`\hobbyrestorepath` This is the user (well, sort of) command that uses the last generated curve.

```

746 \NewDocumentCommand \hobbyrestorepath { m } {
747   \tl_if_exist:cT {g_hobby_#1_path} {
748     \tl_use:c {g_hobby_#1_path}
749   }
750 }

```

(End of definition for \hobbyrestorepath. This function is documented on page ??.)

`\hobbyshowpath` This is the user (well, sort of) command that uses the last generated curve.

```

751 \NewDocumentCommand \hobbyshowpath { m } {
752   \tl_if_exist:cT {g_hobby_#1_path} {
753     \tl_show:c {g_hobby_#1_path}
754   }
755 }

```

(End of definition for \hobbyshowpath. This function is documented on page ??.)

`\hobbygenusepath` This is the user (well, sort of) command that generates a curve and uses it.

```

756 \NewDocumentCommand \hobbygenusepath { } {
757   \array_if_empty:NF \g__hobby_points_array {
758     \hobby_gen_path:
759     \hobby_use_path:
760   }
761 }

```

(End of definition for \hobbygenusepath. This function is documented on page ??.)

`\hobbyclearpath` This is the user (well, sort of) command that generates a curve and uses it.

```

762 \NewDocumentCommand \hobbyclearpath { } {
763   \hobby_clear_path:
764 }

```

(End of definition for \hobbyclearpath. This function is documented on page ??.)

`\hobby_use_path:` This is the command that uses the curve. As the curve data is stored globally, the same data can be reused by calling this function more than once without calling the generating function.

```

765 \tl_new:N \l_tmpc_tl
766 \tl_new:N \l_tmpd_tl
767 \cs_new:Nn \hobby_use_path: {
768   \bool_if:NT \g__hobby_disjoint_bool {
769     \array_get:NnN \g__hobby_points_array {0} \l_tmpa_tl
770     \hobby_moveto:nnV {} {} \l_tmpa_tl
771   }
772   \int_step_inline:nnnn {1} {1} {\g__hobby_npoints_int} {
773     \array_get:NnN \g__hobby_controla_array {##1} \l_tmpa_tl
774     \array_get:NnN \g__hobby_controlb_array {##1} \l_tmpb_tl
775     \array_get:NnN \g__hobby_points_array {##1} \l_tmpc_tl
776     \array_get:NnN \g__hobby_actions_array {##1} \l_tmpd_tl
777     \int_compare:nNnTF {\tl_item:Nn \l_tmpd_tl {1}} = {\g__hobby_draw_int} {
778       \hobby_curveto:VVV \l_tmpa_tl \l_tmpb_tl \l_tmpc_tl
779     }{
780       \bool_gset_false:N \g__hobby_closed_bool
781       \hobby_moveto:VVV \l_tmpa_tl \l_tmpb_tl \l_tmpc_tl
782     }

```

```

783     \tl_if_eq:xnTF {\tl_item:Nn \l_tmpd_tl {2}} {1} {
784         \bool_gset_false:N \g__hobby_closed_bool
785         \hobby_moveto:VVV \l_tmpa_tl \l_tmpb_tl \l_tmpc_tl
786     }{}
787 }
788 \bool_if:NT \g__hobby_closed_bool {
789     \array_get:NnN \g__hobby_points_array {0} \l_tmpa_tl
790     \hobby_close:V \l_tmpa_tl
791 }
792 }

```

(End of definition for \hobby_use_path:. This function is documented on page ??.)

\hobby_save_path:n This command saves all the data needed to reinvoke the curve in a global token list that can be used to restore it afterwards.

```

793 \cs_new:Nn \hobby_save_path:n {
794     \tl_clear:N \l_tmpa_tl
795     \tl_put_right:Nn \l_tmpa_tl {\int_gset:Nn \g__hobby_npoints_int}
796     \tl_put_right:Nx \l_tmpa_tl {{\int_use:N \g__hobby_npoints_int}}
797     \bool_if:NTF \g__hobby_disjoint_bool {
798         \tl_put_right:Nn \l_tmpa_tl {\bool_gset_true:N}
799     }{
800         \tl_put_right:Nn \l_tmpa_tl {\bool_gset_false:N}
801     }
802     \tl_put_right:Nn \l_tmpa_tl {\g__hobby_disjoint_bool}
803     \bool_if:NTF \g__hobby_closed_bool {
804         \tl_put_right:Nn \l_tmpa_tl {\bool_gset_true:N}
805     }{
806         \tl_put_right:Nn \l_tmpa_tl {\bool_gset_false:N}
807     }
808     \tl_put_right:Nn \l_tmpa_tl {\g__hobby_closed_bool}
809     \tl_put_right:Nn \l_tmpa_tl {\array_gclear:N \g__hobby_points_array}
810     \array_map_inline:Nn \g__hobby_points_array {
811         \tl_put_right:Nn \l_tmpa_tl {
812             \array_gput:Nnn \g__hobby_points_array {##1} {##2}
813         }
814     }
815     \tl_put_right:Nn \l_tmpa_tl {\array_gclear:N \g__hobby_actions_array}
816     \array_map_inline:Nn \g__hobby_actions_array {
817         \tl_put_right:Nn \l_tmpa_tl {
818             \array_gput:Nnn \g__hobby_actions_array {##1} {##2}
819         }
820     }
821     \tl_put_right:Nn \l_tmpa_tl {\array_gclear:N \g__hobby_controla_array}
822     \array_map_inline:Nn \g__hobby_controla_array {
823         \tl_put_right:Nn \l_tmpa_tl {
824             \array_gput:Nnn \g__hobby_controla_array {##1} {##2}
825         }
826     }
827     \tl_put_right:Nn \l_tmpa_tl {\array_gclear:N \g__hobby_controlb_array}
828     \array_map_inline:Nn \g__hobby_controlb_array {
829         \tl_put_right:Nn \l_tmpa_tl {
830             \array_gput:Nnn \g__hobby_controlb_array {##1} {##2}
831         }
832     }
833     \tl_gclear_new:c {g_hobby_#1_path}
834     \tl_gset_eq:cN {g_hobby_#1_path} \l_tmpa_tl
835 }

```

(End of definition for \hobby_save_path:n. This function is documented on page ??.)

bby_save_path_to_aux:n

```

836 \int_set:Nn \l_tmpa_int {\char_value_catcode:n {'@}}
837 \char_set_catcode_letter:N @
838 \cs_new:Npn \hobby_save_path_to_aux:n #1 {
839   \bool_if:nT {
840     \tl_if_exist_p:c {g_hobby_#1_path}
841     &&
842     ! \tl_if_exist_p:c {g_hobby_#1_path_saved}
843     &&
844     \l_hobby_save_aux_bool
845   }
846   {
847     \tl_clear:N \l_tmpa_tl
848     \tl_put_right:Nn \l_tmpa_tl {
849       \ExplSyntaxOn
850       \tl_gclear_new:c {g_hobby_#1_path}
851       \tl_gput_right:cn {g_hobby_#1_path}
852     }
853     \tl_put_right:Nx \l_tmpa_tl {
854       {\tl_to_str:c {g_hobby_#1_path}}
855     }
856     \tl_put_right:Nn \l_tmpa_tl {
857       \ExplSyntaxOff
858     }
859     \protected@write\@auxout{}\{
860       \tl_to_str:N \l_tmpa_tl
861     }
862     \tl_new:c {g_hobby_#1_path_saved}
863   }
864 }
865 \char_set_catcode:nn {'@} {\l_tmpa_int}
866 \cs_generate_variant:Nn \hobby_save_path_to_aux:n {x}

```

(End of definition for \hobby_save_path_to_aux:n. This function is documented on page ??.)

\hobby_clear_path:

```

867 \cs_new:Nn \hobby_clear_path:
868 {
869   \array_gclear:N \g__hobby_points_array
870   \array_gclear:N \g__hobby_points_x_array
871   \array_gclear:N \g__hobby_points_y_array
872   \array_gclear:N \g__hobby_angles_array
873   \array_gclear:N \g__hobby_actions_array
874   \array_gclear:N \g__hobby_distances_array
875   \array_gclear:N \g__hobby_tension_out_array
876   \array_gclear:N \g__hobby_tension_in_array
877   \array_gclear:N \g__hobby_excess_angle_array
878   \array_gclear:N \g__hobby_matrix_a_array
879   \array_gclear:N \g__hobby_matrix_b_array
880   \array_gclear:N \g__hobby_matrix_c_array
881   \array_gclear:N \g__hobby_matrix_d_array
882   \array_gclear:N \g__hobby_vector_u_array
883   \array_gclear:N \g__hobby_psi_array
884   \array_gclear:N \g__hobby_theta_array
885   \array_gclear:N \g__hobby_phi_array
886   \array_gclear:N \g__hobby_sigma_array
887   \array_gclear:N \g__hobby_rho_array
888   \array_gclear:N \g__hobby_controla_array
889   \array_gclear:N \g__hobby_controlb_array
890   \bool_gset_false:N \g__hobby_closed_bool
891   \bool_gset_false:N \g__hobby_disjoint_bool

```

```

892
893 \int_gset:Nn \g__hobby_npoints_int {-1}
894 \int_gset:Nn \g__hobby_draw_int {1}
895 \fp_gset_eq:NN \g__hobby_in_angle_fp \c_inf_fp
896 \fp_gset_eq:NN \g__hobby_out_angle_fp \c_inf_fp
897 \fp_gset_eq:NN \g__hobby_in_curl_fp \c_one_fp
898 \fp_gset_eq:NN \g__hobby_out_curl_fp \c_one_fp
899 }

```

(End of definition for `\hobby_clear_path`:. This function is documented on page ??.)

```

900 \ExplSyntaxOff

```

1.2 PGF Library

The PGF level is very simple. All we do is set up the path-construction commands that get passed to the path-generation function.

```

901 \input{hobby.code.tex}

```

Points are communicated as key-pairs. These keys translate from the L^AT_EX3 style points to PGF points.

```

902 \pgfkeys{
903   /pgf/hobby/.is family,
904   /pgf/hobby/.cd,
905   point/.code={%
906     \hobby@parse@pt#1\relax}
907 }
908 \def\hobby@parse@pt#1,#2\relax{%
909   \pgf@x=#1cm\relax
910   \pgf@y=#2cm\relax
911 }

```

hobbyatan2 The original PGF version of `atan2` had the arguments the wrong way around. This was fixed in the CVS version in July 2013, but as old versions are likely to be in use for some time, we define a wrapper function that ensures that the arguments are correct.

```

912 \pgfmathparse{atan2(0,1)}
913 \def\hobby@temp{0.0}
914 \ifx\pgfmathresult\hobby@temp
915   \pgfmathdeclarefunction{hobbyatan2}{2}{%
916     \pgfmathatantwo@{#1}{#2}%
917   }
918 \else
919   \pgfmathdeclarefunction{hobbyatan2}{2}{%
920     \pgfmathatantwo@{#2}{#1}%
921   }
922 \fi

```

(End of definition for `hobbyatan2`. This function is documented on page ??.)

\hobby@curveto This is passed to the path-generation code to translate the path into a PGF path.

```

923 \def\hobby@curveto#1#2#3{%
924   \pgfpathcurveto{\hobby@topgf{#1}}{\hobby@topgf{#2}}{\hobby@topgf{#3}}%
925 }

```

(End of definition for `\hobby@curveto`. This function is documented on page ??.)

\hobby@moveto This is passed to the path-generation code to translate the path into a PGF path.

```

926 \def\hobby@moveto#1#2#3{%
927   \pgfpathmoveto{\hobby@topgf{#3}}%
928 }

```

(End of definition for `\hobby@moveto`. This function is documented on page ??.)

`\hobby@topgf` Translates a L^AT_EX3 point to a PGF point.

```

929 \def\hobby@topgf#1{%
930     \pgfqkeys{/pgf/hobby}{point={#1}}%
931 }

```

(End of definition for \hobby@topgf. This function is documented on page ??.)

`\hobby@close` Closes a path.

```

932 \def\hobby@close#1{%
933     \pgfpathclose
934 }

```

(End of definition for \hobby@close. This function is documented on page ??.)

`\pgfpathhobby` Low-level interface to the hobby construction. This sets up the commands and starts the iterator.

```

935 \def\pgfpathhobby{%
936     \pgfutil@ifnextchar\bgroup{\pgfpath@hobby}{\pgfpath@hobby{}}}
937 \def\pgfpath@hobby#1{%
938     \hobbyinit\hobby@moveto\hobby@curveto\hobby@close
939     \hobbysetparams{#1}%
940     \pgfmathsetmacro\hobby@x{\the\pgf@path@lastx/1cm}%
941     \pgfmathsetmacro\hobby@y{\the\pgf@path@lasty/1cm}%
942     \hobbyaddpoint{point={\hobby@x, \hobby@y}}%
943 }

```

(End of definition for \pgfpathhobby. This function is documented on page ??.)

`\pgfpathhobbypt` Adds a point to the construction

```

944 \def\pgfpathhobbypt#1{%
945     #1%
946     \pgfmathsetmacro\hobby@x{\the\pgf@x/1cm}%
947     \pgfmathsetmacro\hobby@y{\the\pgf@y/1cm}%
948     \pgfutil@ifnextchar\bgroup{\pgfpathhobbyptparams}{\pgfpathhobbyptparams{}}%
949 }

```

(End of definition for \pgfpathhobbypt. This function is documented on page ??.)

`\pgfpathhobbyptparams`

```

950 \def\pgfpathhobbyptparams#1{%
951     \hobbyaddpoint{#1,point={\hobby@x, \hobby@y}}%
952 }

```

(End of definition for \pgfpathhobbyptparams. This function is documented on page ??.)

`\pgfpathhobbyend`

```

953 \def\pgfpathhobbyend{%
954     \ifhobby@externalise
955         \ifx\hobby@path@name\pgfutil@empty
956             \hobbygenusepath
957         \else
958             \hobbygenuseifnecpath{\hobby@path@name}%
959         \fi
960     \else
961         \hobbygenusepath
962     \fi
963     \ifx\hobby@path@name\pgfutil@empty
964     \else
965         \hobbysavepath{\hobby@path@name}%
966     \fi
967     \global\let\hobby@path@name=\pgfutil@empty
968 }

```


(End of definition for \pgfpathhobbyend. This function is documented on page ??.)

Plot handlers

\pgfplothanderhobby Basic plot handler; uses full algorithm but therefore expensive

```

969 \def\pgfplothanderhobby{%
970   \def\pgf@plotstreamstart{%
971     \hobbyinit\hobby@moveto\hobby@curveto\hobby@close
972     \global\let\pgf@plotstreampoint=\pgf@plot@hobby@firstpt
973     \global\let\pgf@plotstreamspecial=\pgfutil@gobble
974   \gdef\pgf@plotstreamend{%
975     \ifhobby@externalise
976       \ifx\hobby@path@name\pgfutil@empty
977         \hobbygenusepath
978       \else
979         \hobbygenuseifnecpath{\hobby@path@name}%
980       \fi
981     \else
982       \hobbygenusepath
983     \fi
984     \ifx\hobby@path@name\pgfutil@empty
985     \else
986       \hobbysavepath{\hobby@path@name}%
987     \fi
988     \global\let\hobby@path@name=\pgfutil@empty
989   }%
990   \let\tikz@scan@point@options=\pgfutil@empty
991 }
992 }
```

(End of definition for \pgfplothanderhobby. This function is documented on page ??.)

\pgfplothandlerclosedhobby Same as above but produces a closed curve

```

993 \def\pgfplothandlerclosedhobby{%
994   \def\pgf@plotstreamstart{%
995     \hobbyinit\hobby@moveto\hobby@curveto\hobby@close
996     \hobbysetparams{closed=true,disjoint=true}%
997     \global\let\pgf@plotstreampoint=\pgf@plot@hobby@firstpt
998     \global\let\pgf@plotstreamspecial=\pgfutil@gobble
999   \gdef\pgf@plotstreamend{%
1000     \ifhobby@externalise
1001       \ifx\hobby@path@name\pgfutil@empty
1002         \hobbygenusepath
1003       \else
1004         \hobbygenuseifnecpath{\hobby@path@name}%
1005       \fi
1006     \else
1007       \hobbygenusepath
1008     \fi
1009     \ifx\hobby@path@name\pgfutil@empty
1010     \else
1011       \hobbysavepath{\hobby@path@name}%
1012     \fi
1013     \global\let\hobby@path@name=\pgfutil@empty
1014   }%
1015 }
1016 }
```

(End of definition for \pgfplothandlerclosedhobby. This function is documented on page ??.)

\pgf@plot@hobby@firstpt First point, move or line as appropriate and then start the algorithm.

```

1017 \def\pgf@plot@hobby@firstpt#1{%
```

```

1018 \pgf@plot@first@action{#1}%
1019 \pgf@plot@hobby@handler{#1}%
1020 \global\let\pgf@plot@streampoint=\pgf@plot@hobby@handler
1021 }

```

(End of definition for \pgf@plot@hobby@firstpt. This function is documented on page ??.)

`\pgf@plot@hobby@handler` Add points to the array for the algorithm to work on.

```

1022 \def\pgf@plot@hobby@handler#1{%
1023   #1%
1024   \pgfmathsetmacro\hobby@x{\the\pgf@x/1cm}%
1025   \pgfmathsetmacro\hobby@y{\the\pgf@y/1cm}%
1026   \hobbyaddpoint{point={\hobby@x, \hobby@y}}%
1027 }

```

(End of definition for \pgf@plot@hobby@handler. This function is documented on page ??.)

`\pgfplothandlerquickhobby` Uses the “quick” algorithm.

```

1028 \def\pgfplothandlerquickhobby{%
1029   \def\pgf@plot@streamstart{%
1030     \global\let\hobby@quick@curveto=\pgfpathcurveto
1031     \global\let\pgf@plot@streampoint=\pgf@plot@qhobby@firstpt
1032     \global\let\pgf@plot@streamspecial=\pgfutil@gobble
1033     \global\let\pgf@plot@streamend=\pgf@plot@qhobby@end
1034   }
1035 }

```

(End of definition for \pgfplothandlerquickhobby. This function is documented on page ??.)

`\pgf@plot@qhobby@firstpt` Carry out first action (move or line) and save point.

```

1036 \def\pgf@plot@qhobby@firstpt#1{%
1037   #1%
1038   \edef\hobby@temp{\noexpand\pgf@plot@first@action{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}\hobby@temp}%
1039   \xdef\hobby@qpoints{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1040   \gdef\hobby@qpointa{%
1041     \gdef\hobby@angle{%
1042       \global\let\pgf@plot@streampoint=\pgf@plot@qhobby@secondpt
1043     }

```

(End of definition for \pgf@plot@qhobby@firstpt. This function is documented on page ??.)

`\pgf@plot@qhobby@secondpt` Also need to save second point.

```

1044 \def\pgf@plot@qhobby@secondpt#1{%
1045   #1%
1046   \xdef\hobby@qpointa{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1047   \global\let\pgf@plot@streampoint=\pgf@plot@qhobby@handler
1048 }

```

(End of definition for \pgf@plot@qhobby@secondpt. This function is documented on page ??.)

`\pgf@plot@qhobby@handler` Wrapper around the computation macro that saves the variables globally.

```

1049 \def\pgf@plot@qhobby@handler#1{%
1050   #1
1051   \edef\hobby@temp{\noexpand\hobby@quick@compute{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}\hobby@temp}%
1052   \global\let\hobby@qpointa=\hobby@qpointa
1053   \global\let\hobby@qpoints=\hobby@qpoints
1054   \global\let\hobby@angle=\hobby@angle

```

Also need to save some data for the last point

```

1055   \global\let\hobby@thetaone=\hobby@thetaone
1056   \global\let\hobby@phitwo=\hobby@phitwo
1057   \global\let\hobby@done=\hobby@done
1058   \global\let\hobby@omegaone=\hobby@omegaone
1059 }

```

(End of definition for \pgf@plot@qhobby@handler. This function is documented on page ??.)

\pgf@plot@qhobby@end Wrapper around the finalisation step.

```
1060 \def\pgf@plot@qhobby@end{%
1061   \hobby@quick@computeend
1062 }
```

(End of definition for \pgf@plot@qhobby@end. This function is documented on page ??.)

\hobby@sf Working with points leads to computations out of range so we scale to get them into the computable arena.

```
1063 \pgfmathsetmacro\hobby@sf{10cm}
```

(End of definition for \hobby@sf. This function is documented on page ??.)

\hobby@quick@compute This is the macro that does all the work of computing the control points. The argument is the current point, \hobby@qpointa is the middle point, and \hobby@qpoints is the first point.

```
1064 \def\hobby@quick@compute#1{%
```

Save the current (second - counting from zero) point in \pgf@xb and \pgf@yb.

```
1065   #1%
1066   \pgf@xb=\pgf@x
1067   \pgf@yb=\pgf@y
```

Save the previous (first) point in \pgf@xa and \pgf@ya.

```
1068   \hobby@qpointa
1069   \pgf@xa=\pgf@x
1070   \pgf@ya=\pgf@y
```

Adjust so that (\pgf@xb,\pgf@yb) is the vector from second to third. Then compute and store the distance and angle of this vector. We view this as the vector *from* the midpoint and everything to do with that point has the suffix one. Note that we divide by the scale factor here.

```
1071   \advance\pgf@xb by -\pgf@xa
1072   \advance\pgf@yb by -\pgf@ya
1073   \pgfmathsetmacro\hobby@done{sqrt((\pgf@xb/\hobby@sf)^2 + (\pgf@yb/\hobby@sf)^2)}%
1074   \pgfmathsetmacro\hobby@omegaone{rad(hobbyatan2(\pgf@yb,\pgf@xb))}%
```

Now we do the same with the vector from the zeroth to the first point.

```
1075   \hobby@qpoints
1076   \advance\pgf@xa by -\pgf@x
1077   \advance\pgf@ya by -\pgf@y
1078   \pgfmathsetmacro\hobby@dzero{sqrt((\pgf@xa/\hobby@sf)^2 + (\pgf@ya/\hobby@sf)^2)}%
1079   \pgfmathsetmacro\hobby@omegazero{rad(hobbyatan2(\pgf@ya,\pgf@xa))}%
```

\hobby@psi is the angle subtended at the midpoint. We adjust to ensure that it is in the right range.

```
1080   \pgfmathsetmacro\hobby@psi{\hobby@omegaone - \hobby@omegazero}%
1081   \pgfmathsetmacro\hobby@psi{\hobby@psi > pi ? \hobby@psi - 2*pi : \hobby@psi}%
1082   \pgfmathsetmacro\hobby@psi{\hobby@psi < -pi ? \hobby@psi + 2*pi : \hobby@psi}%
```

Now we test to see if we're on the first run or not. If the first, we have no incoming angle.

```
1083   \ifx\hobby@angle\pgfutil@empty
```

First.

```
1084   \pgfmathsetmacro\hobby@thetaone{-\hobby@psi * \hobby@done%
1085   /(\hobby@done + \hobby@dzero)}%
1086   \pgfmathsetmacro\hobby@thetazero{-\hobby@psi - \hobby@thetaone}%
1087   \let\hobby@phone=\hobby@thetazero
1088   \let\hobby@phitwo=\hobby@thetaone
1089   \else
```

Second or later.

```

1090 \let\hobby@thetazero=\hobby@angle
1091 \pgfmathsetmacro\hobby@thetaone{%
1092 -(2 * \hobby@psi + \hobby@thetazero) * \hobby@done%
1093 / (2 * \hobby@done + \hobby@dzero)}%
1094 \pgfmathsetmacro\hobby@phione{-\hobby@psi - \hobby@thetaone}%
1095 \let\hobby@phitwo=\hobby@thetaone
1096 \fi

```

Save the outgoing angle.

```

1097 \let\hobby@angle=\hobby@thetaone

```

Compute the control points from the angles.

```

1098 \hobby@quick@ctrlpts{\hobby@thetazero}{\hobby@phione}{\hobby@qpoints}{\hobby@qpointa}{\hobby@dzero}

```

Now call the call-back function

```

1099 \edef\hobby@temp{\noexpand\hobby@quick@curveto{\noexpand\pgfqpoint{\the\pgf@xa}{\the\pgf@ya}}{\noexpand\hobby@temp}
1100 \hobby@temp

```

Cycle the points round for the next iteration.

```

1101 \global\let\hobby@qpoints=\hobby@qpointa
1102 #1
1103 \xdef\hobby@qpointa{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%

```

Save needed values in global macros

```

1104 \global\let\hobby@angle=\hobby@angle
1105 \global\let\hobby@phitwo=\hobby@phitwo
1106 \global\let\hobby@thetaone=\hobby@thetaone
1107 \global\let\hobby@done=\hobby@done
1108 \global\let\hobby@omegaone=\hobby@omegaone
1109 }

```

(End of definition for \hobby@quick@compute. This function is documented on page ??.)

`\hobby@quick@computeend` This is the additional code for the final run.

```

1110 \def\hobby@quick@computeend{%

```

Compute the control points for the second part of the curve and add that to the path.

```

1111 \hobby@quick@ctrlpts{\hobby@thetaone}{\hobby@phitwo}{\hobby@qpoints}{\hobby@qpointa}{\hobby@done}

```

Now call the call-back function

```

1112 \edef\hobby@temp{\noexpand\hobby@quick@curveto{\noexpand\pgfqpoint{\the\pgf@xa}{\the\pgf@ya}}{\noexpand\hobby@temp}
1113 \hobby@temp
1114 }%

```

(End of definition for \hobby@quick@computeend. This function is documented on page ??.)

`\hobby@quick@ctrlpts` Compute the control points from the angles and points given.

```

1115 \def\hobby@quick@ctrlpts#1#2#3#4#5#6{%
1116 \pgfmathsetmacro\hobby@alpha{%
1117 sqrt(2) * (sin(#1 r) - 1/16 * sin(#2 r))%
1118 * (sin(#2 r) - 1/16 * sin(#1 r))%
1119 * (cos(#1 r) - cos(#2 r))}%
1120 \pgfmathsetmacro\hobby@rho{%
1121 (2 + \hobby@alpha)/(1 + (1 - (3 - sqrt(5))/2))%
1122 * cos(#1 r) + (3 - sqrt(5))/2 * cos(#2 r))}%
1123 \pgfmathsetmacro\hobby@sigma{%
1124 (2 - \hobby@alpha)/(1 + (1 - (3 - sqrt(5))/2))%
1125 * cos(#2 r) + (3 - sqrt(5))/2 * cos(#1 r))}%
1126 #3%
1127 \pgf@xa=\pgf@x
1128 \pgf@ya=\pgf@y
1129 \pgfmathsetlength\pgf@xa{%
1130 \pgf@xa + #5 * \hobby@rho%

```

```

1131 * cos((#1 + #6) r)/3*\hobby@sf}%
1132 \pgfmathsetlength\pgf@ya{%
1133   \pgf@ya + #5 * \hobby@rho%
1134 * sin((#1 + #6) r)/3*\hobby@sf}%
1135 #4%
1136 \pgf@xb=\pgf@x
1137 \pgf@yb=\pgf@y
1138 \pgfmathsetlength\pgf@xb{%
1139   \pgf@xb - #5 * \hobby@sigma%
1140 * cos((-#2 + #6) r)/3*\hobby@sf}%
1141 \pgfmathsetlength\pgf@yb{%
1142   \pgf@yb - #5 * \hobby@sigma%
1143 * sin((-#2 + #6) r)/3*\hobby@sf}%
1144 #4%
1145 }

```

(End of definition for \hobby@quick@ctrlpts. This function is documented on page ??.)

1.3 TikZ Library

```

1146 \usepgflibrary{hobby}
1147 \let\hobby@this@opts=\pgfutil@empty
1148 \let\hobby@next@opts=\pgfutil@empty
1149 \let\hobby@action=\pgfutil@empty
1150 \let\hobby@path@name=\pgfutil@empty
1151 \newif\ifhobby@externalise

```

We set various TikZ keys. These include the `to path` constructor and all the various parameters that will eventually get passed to the path-generation code.

```

1152 \def\hobby@point@options{%
1153 \tikzset{
1154   curve through/.style={
1155     to path={
1156       \pgfextra{
1157         \expandafter\curvethrough\expandafter[\hobby@next@opts]{%
1158           (\tikztostart) .. #1 .. (\tikztotarget)%
1159         }
1160       }
1161     }
1162   },
1163   tension in/.code = {%
1164     \expandafter\gdef\expandafter\hobby@point@options\expandafter%
1165     {\hobby@point@options,tension in=#1}%
1166   },
1167   tension out/.code = {%
1168     \expandafter\gdef\expandafter\hobby@point@options\expandafter%
1169     {\hobby@point@options,tension out=#1}%
1170   },
1171   tension/.append code = {%
1172     \expandafter\gdef\expandafter\hobby@point@options\expandafter%
1173     {\hobby@point@options,tension=#1}%
1174   },
1175   excess angle/.code = {%
1176     \expandafter\gdef\expandafter\hobby@point@options\expandafter%
1177     {\hobby@point@options,excess angle=#1}%
1178   },
1179   break/.code = {%
1180     \expandafter\gdef\expandafter\hobby@point@options\expandafter%
1181     {\hobby@point@options,break=#1}%
1182   },
1183   blank/.code = {%
1184     \expandafter\gdef\expandafter\hobby@point@options\expandafter%

```

```

1185     {\hobby@point@options,blank=#1}%
1186 },
1187 designated Hobby path/.initial={next},
1188 clear next Hobby path options/.code={%
1189     \gdef\hobby@next@opts{%
1190 },
1191 clear this Hobby path options/.code={%
1192     \gdef\hobby@this@opts{%
1193 },
1194 clear Hobby path options/.style={%
1195     clear \pgfkeysvalueof{/tikz/designated Hobby path} Hobby path options
1196 },
1197 add option to this Hobby path/.code={%
1198     \expandafter\gdef\expandafter\hobby@this@opts\expandafter{\hobby@this@opts#1,}%
1199 },
1200 add option to next Hobby path/.code={%
1201     \expandafter\gdef\expandafter\hobby@next@opts\expandafter{\hobby@next@opts#1,}%
1202 },
1203 add option to Hobby path/.style={%
1204     add option to \pgfkeysvalueof{/tikz/designated Hobby path} Hobby path={#1}%
1205 },
1206 closed/.style = {%
1207     add option to Hobby path={closed=#1,disjoint=#1}%
1208 },
1209 invert blank/.style = {%
1210     add option to Hobby path={invert blank=#1}%
1211 },
1212 closed/.default = true,
1213 blank/.default = true,
1214 break/.default = true,
1215 invert blank/.default = true,
1216 in angle/.code = {%
1217     \pgfmathparse{(#1)*pi/180}%
1218     \edef\@temp{in angle=\pgfmathresult,}%
1219     \pgfkeysalso{add option to Hobby path/.expand once=\@temp}%
1220 },
1221 out angle/.code = {%
1222     \pgfmathparse{(#1)*pi/180}%
1223     \edef\@temp{out angle=\pgfmathresult,}%
1224     \pgfkeysalso{add option to Hobby path/.expand once=\@temp}%
1225 },
1226 in curl/.style = {%
1227     add option to Hobby path={in curl=#1}%
1228 },
1229 out curl/.style = {%
1230     add option to Hobby path={out curl=#1}%
1231 },
1232 use Hobby shortcut/.code={%
1233     \let\tikz@curveto@auto=\hobby@curveto@override
1234     \global\let\hobby@curveto@delegate=\hobby@curveto@auto
1235 },
1236 use quick Hobby shortcut/.code={%
1237     \let\tikz@curveto@auto=\hobby@curveto@override
1238     \global\let\hobby@curveto@delegate=\hobby@qcurveto@auto
1239 },
1240 use previous Hobby path/.code={%
1241     \hobbyusepath{#1}%
1242 },
1243 use previous Hobby path/.default={},%
1244 save Hobby path/.code={%
1245     \xdef\hobby@path@name{#1}%

```

```

1246 },
1247 restore Hobby path/.code={%
1248   \hobbyinit\hobby@tikz@moveto\hobby@tikz@curveto\hobby@tikz@close
1249   \global\let\hobby@collected@onpath\pgfutil@empty
1250   \hobbyrestorepath{#1}%
1251 },
1252 restore and use Hobby path/.code 2 args={%
1253   \hobbyinit\hobby@tikz@moveto\hobby@tikz@curveto\hobby@tikz@close
1254   \global\let\hobby@collected@onpath\pgfutil@empty
1255   \hobbyrestorepath{#1}%
1256   \hobbyusepath{#2}%
1257 },
1258 show Hobby path/.code={%
1259   \hobbyshowpath{#1}%
1260 },
1261 Hobby action/.code={%
1262   \expandafter\gdef\expandafter\hobby@action\expandafter{\hobby@action#1}%
1263 },
1264 Hobby finish/.style={%
1265   Hobby action=\hobby@finish%
1266 },
1267 Hobby externalise/.is if=hobby@externalise,
1268 Hobby externalize/.is if=hobby@externalise
1269 }

```

`\hobby@tikz@curveto` This is passed to the path-generation code to translate the path into a PGF path.

```

1270 \def\hobby@tikz@curveto#1#2#3{%
1271   \pgfutil@ifundefined{tikz@timer@start}{%
1272     \expandafter\hobby@topgf\expandafter{\hobby@initial@pt}%
1273     \edef\tikz@timer@start{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1274   }{%
1275     \hobby@topgf{#1}%
1276     \edef\tikz@timer@cont@one{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1277     \hobby@topgf{#2}%
1278     \edef\tikz@timer@cont@two{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1279     \hobby@topgf{#3}%
1280     \let\tikz@timer=\tikz@timer@curve
1281     \edef\tikz@timer@end{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1282     \ifx\hobby@collected@onpath\pgfutil@empty
1283       \else
1284         \expandafter\hobby@nodes@onpath\hobby@collected@onpath\relax\relax
1285       \fi
1286       \pgfpathcurveto{\hobby@topgf{#1}}{\hobby@topgf{#2}}{\hobby@topgf{#3}}%
1287       \hobby@topgf{#3}%
1288       \edef\tikz@timer@start{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1289     }

```

(End of definition for \hobby@tikz@curveto. This function is documented on page ??.)

`\hobby@tikz@moveto` This is passed to the path-generation code to translate the path into a PGF path.

```

1290 \def\hobby@tikz@moveto#1#2#3{%
1291   \pgfutil@ifundefined{tikz@timer@start}{%
1292     \expandafter\hobby@topgf\expandafter{\hobby@initial@pt}%
1293     \edef\tikz@timer@start{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1294   }{%
1295     \hobby@topgf{#3}%
1296     \edef\tikz@timer@end{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1297     \def\pgf@temp{#1}%
1298     \ifx\pgf@temp\pgfutil@empty
1299       \let\tikz@timer=\tikz@timer@line
1300     \expandafter\def\expandafter\hobby@collected@onpath\expandafter{\expandafter\hobby@collected@onpath\expandafter}
1301   }

```

```

1302 \hobby@topgf{#1}%
1303 \edef\tikz@timer@cont@one{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1304 \hobby@topgf{#2}%
1305 \edef\tikz@timer@cont@two{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1306 \let\tikz@timer=\tikz@timer@curve
1307 \fi
1308 \ifx\hobby@collected@onpath\pgfutil@empty
1309 \else
1310 \expandafter\hobby@nodes@onpath\hobby@collected@onpath\relax\relax
1311 \fi
1312 \pgfpathmoveto{\hobby@topgf{#3}}%
1313 \hobby@topgf{#3}%
1314 \edef\tikz@timer@start{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1315 }

```

(End of definition for \hobby@tikz@moveto. This function is documented on page ??.)

\hobby@tikz@close Closes a path.

```

1316 \def\hobby@tikz@close#1{%
1317 \hobby@topgf{#1}%
1318 \edef\tikz@timer@end{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1319 \let\tikz@timer=\tikz@timer@line
1320 \ifx\hobby@collected@onpath\pgfutil@empty
1321 \else
1322 \expandafter\hobby@nodes@onpath\hobby@collected@onpath\relax\relax
1323 \fi
1324 \pgfpathclose
1325 }

```

(End of definition for \hobby@tikz@close. This function is documented on page ??.)

\hobby@nodes@onpath

```

1326 \def\hobby@nodes@onpath#1#2\relax{%
1327 \gdef\hobby@collected@onpath{#2}%
1328 \def\pgf@temp{#1}%
1329 \ifx\pgf@temp\pgfutil@empty
1330 \else
1331 \def\@gtempa{\relax}
1332 \ifx\pgf@temp\@gtempa
1333 \else
1334 \tikz@node@is@a@labeltrue
1335 \tikz@scan@next@command#1\pgf@stop
1336 \tikz@node@is@a@labelfalse
1337 \fi
1338 \fi
1339 }

```

(End of definition for \hobby@nodes@onpath. This function is documented on page ??.)

\curvethrough This is the parent command. We initialise the path-generation code, set any parameters, and then hand over control to the point processing macro.

```

1340 \newcommand\curvethrough[2][]{%
1341 \hobbyinit\hobby@tikz@moveto\hobby@tikz@curveto\hobby@tikz@close
1342 \global\let\hobby@collected@onpath\pgfutil@empty
1343 \let\hobby@initial@pt\pgfutil@empty
1344 \hobbysetparams{#1}%
1345 \tikzset{designated Hobby path=this}%
1346 \global\let\hobby@this@opts=\pgfutil@empty
1347 \global\let\hobby@next@opts=\pgfutil@empty
1348 \let\tikz@scan@point@options=\pgfutil@empty
1349 \def\hobby@point@options{%
1350 \tikz@scan@one@point\hobby@processpt #2 \relax%
1351 }

```


(End of definition for \curvethrough. This function is documented on page ??.)

\hobby@processpt This processes a list of points in the format (0,0) [...] (1,1). Each point is scanned by TikZ and then added to the stack to be built into the path. If there are any remaining points, we call ourself again with them. Otherwise, we hand over control to the path-generation code.

```

1352 \newcommand\hobby@processpt[1]{%
1353   #1%
1354   \pgfmathsetmacro\hobby@x{\the\pgf@x/1cm}%
1355   \pgfmathsetmacro\hobby@y{\the\pgf@y/1cm}%
1356   \ifx\hobby@initial@pt\pgfutil@empty
1357     \xdef\hobby@initial@pt{\hobby@x, \hobby@y}%
1358   \fi
1359   \expandafter\hobbyaddpoint\expandafter{\hobby@point@options,%
1360     point={\hobby@x, \hobby@y}}%
1361   \def\hobby@point@options{}%
1362   \let\tikz@scan@point@options=\pgfutil@empty
1363   \pgfutil@ifnextchar\relax{%
1364     \expandafter\hobbysetparams\expandafter{\hobby@this@opts}%
1365   \ifhobby@externalise
1366     \ifx\hobby@path@name\pgfutil@empty
1367       \hobbygenusepath
1368     \else
1369       \hobbygenuseifnecpath{\hobby@path@name}%
1370     \fi
1371   \else
1372     \hobbygenusepath
1373   \fi
1374   \ifx\hobby@path@name\pgfutil@empty
1375   \else
1376     \hobbysavepath{\hobby@path@name}%
1377   \fi
1378   \global\let\hobby@path@name=\pgfutil@empty
1379 }{%
1380   \pgfutil@ifnextchar.{%
1381     \hobby@swallowdots}{%
1382       \tikz@scan@one@point\hobby@processpt}}}
```

(End of definition for \hobby@processpt. This function is documented on page ??.)

\hobby@swallowdots Remove dots from the input stream.

```

1383 \def\hobby@swallowdots.{%
1384   \pgfutil@ifnextchar.{%
1385     \hobby@swallowdots}{%
1386       \tikz@scan@one@point\hobby@processpt}}
```

(End of definition for \hobby@swallowdots. This function is documented on page ??.)

There is a “spare hook” in the TikZ path processing code. If TikZ encounters a path of the form (0,0) .. (1,1) then it calls a macro \tikz@curveto@auto. However, that macro is not defined in the TikZ code. The following code provides a suitable definition. To play nice, we don’t install it by default but define a key (defined above) that installs it.

\hobby@curveto@override

```

1387 \def\hobby@curveto@override{%
1388   \hobby@curveto@delegate}
```

(End of definition for \hobby@curveto@override. This function is documented on page ??.)

\hobby@curveto@auto When we’re called by TikZ, we initialise the path generation code and start adding points. To ensure that the generation code is called, we add a lot of hooks to lots of TikZ commands.

```

1389 \def\hobby@curveto@auto{%
1390   \hobbyinit\hobby\tikz@moveto\hobby\tikz@curveto\hobby\tikz@close
```

```

1391 \expandafter\gdef\expandafter\hobby@collected@onpath\expandafter{\expandafter\tikz@collected@onpath}
1392 \let\tikz@collected@onpath=\pgfutil@empty
1393 \pgfmathsetmacro\hobby@x{\the\tikz@lastx/1cm}%
1394 \pgfmathsetmacro\hobby@y{\the\tikz@lasty/1cm}%
1395 \xdef\hobby@initial@pt{\hobby@x, \hobby@y}%
1396 \expandafter\hobbysetparams\expandafter{\hobby@next@opts}%
1397 \expandafter\hobbyaddpoint\expandafter{\hobby@point@options,%
1398   point={\hobby@x, \hobby@y} }%
1399 \hobby@init@tikz@commands
1400 \tikzset{designated Hobby path=this}%
1401 \let\tikz@scan@point@options=\pgfutil@empty
1402 \global\let\hobby@action=\pgfutil@empty
1403 \global\let\hobby@this@opts=\pgfutil@empty
1404 \global\let\hobby@next@opts=\pgfutil@empty
1405 \global\let\hobby@point@options=\pgfutil@empty
1406 \tikz@scan@one@point\hobby@addfromtikz%
1407 }

```

(End of definition for \hobby@curveto@auto. This function is documented on page ??.)

\hobby@addfromtikz This adds our current point to the stack.

```

1408 \def\hobby@addfromtikz#1{%
1409   #1%
1410   \tikz@make@last@position{#1}%
1411   \pgfmathsetmacro\hobby@x{\the\pgf@x/1cm}%
1412   \pgfmathsetmacro\hobby@y{\the\pgf@y/1cm}%
1413   \expandafter\hobbysetparams\expandafter{\hobby@this@opts}%
1414   \expandafter\hobbyaddpoint\expandafter{\hobby@point@options,%
1415     point={\hobby@x, \hobby@y}}%
1416   \hobby@action
1417   \global\let\hobby@this@opts=\pgfutil@empty
1418   \global\let\hobby@action=\pgfutil@empty
1419   \global\let\hobby@point@options=\pgfutil@empty
1420   \tikz@scan@next@command%
1421 }

```

(End of definition for \hobby@addfromtikz. This function is documented on page ??.)

\hobby@init@tikz@commands

```

1422 \def\hobby@init@tikz@commands{%
1423   \hobby@init@tikz@modcmd\tikz@movetoabs
1424   \hobby@init@tikz@modcmd\tikz@movetorel
1425   \hobby@init@tikz@modcmd\tikz@lineto
1426   \hobby@init@tikz@modcmd\tikz@rect
1427   \hobby@init@tikz@modcmd\tikz@cchar
1428   \hobby@init@tikz@modcmd\tikz@finish
1429   \hobby@init@tikz@modcmd\tikz@arcA
1430   \hobby@init@tikz@modcmd\tikz@e@char
1431   \hobby@init@tikz@modcmd\tikz@g@char
1432   \hobby@init@tikz@modcmd\tikz@schar
1433   \hobby@init@tikz@modcmd\tikz@vh@lineto
1434   \hobby@init@tikz@modcmd\tikz@pchar
1435   \hobby@init@tikz@modcmd\tikz@to
1436   \hobby@init@tikz@modcmd\pgf@stop
1437   \hobby@init@tikz@modcmd\tikz@decoration
1438   \global\let\hobby@curveto@delegate=\hobby@midcurveto@auto
1439 }

```

(End of definition for \hobby@init@tikz@commands. This function is documented on page ??.)

@restore@tikz@commands

```

1440 \def\hobby@restore@tikz@commands{%
1441   \hobby@restore@tikz@modcmd\tikz@movetoabs
1442   \hobby@restore@tikz@modcmd\tikz@movetorel
1443   \hobby@restore@tikz@modcmd\tikz@lineto
1444   \hobby@restore@tikz@modcmd\tikz@rect
1445   \hobby@restore@tikz@modcmd\tikz@ccchar
1446   \hobby@restore@tikz@modcmd\tikz@finish
1447   \hobby@restore@tikz@modcmd\tikz@arcA
1448   \hobby@restore@tikz@modcmd\tikz@e@char
1449   \hobby@restore@tikz@modcmd\tikz@g@char
1450   \hobby@restore@tikz@modcmd\tikz@schar
1451   \hobby@restore@tikz@modcmd\tikz@vh@lineto
1452   \hobby@restore@tikz@modcmd\tikz@pchar
1453   \hobby@restore@tikz@modcmd\tikz@to
1454   \hobby@restore@tikz@modcmd\pgf@stop
1455   \hobby@restore@tikz@modcmd\tikz@decoration
1456   \global\let\hobby@curveto@delegate=\hobby@curveto@auto
1457 }

```

(End of definition for \hobby@restore@tikz@commands. This function is documented on page ??.)

hobby@init@tikz@modcmd

```

1458 \def\hobby@init@tikz@modcmd#1{%
1459   \expandafter\global\expandafter\let\csname hobby@orig@\string#1\endcsname=#1%
1460   \gdef#1{\hobby@finish#1}%
1461 }

```

(End of definition for \hobby@init@tikz@modcmd. This function is documented on page ??.)

by@restore@tikz@modcmd

```

1462 \def\hobby@restore@tikz@modcmd#1{%
1463   \expandafter\global\expandafter\let\expandafter#1\csname hobby@orig@\string#1\endcsname%
1464 }

```

(End of definition for \hobby@restore@tikz@modcmd. This function is documented on page ??.)

\hobby@midcurveto@auto

```

1465 \def\hobby@midcurveto@auto{%
1466   \expandafter\expandafter\expandafter\gdef\expandafter\expandafter\expandafter\hobby@collected@onpath=
1467   \let\tikz@collected@onpath=\pgfutil@empty
1468   \let\tikz@scan@point@options=\pgfutil@empty
1469   \global\let\hobby@action=\pgfutil@empty
1470   \global\let\hobby@this@opts=\pgfutil@empty
1471   \global\let\hobby@point@options=\pgfutil@empty
1472   \tikz@scan@one@point\hobby@addfromtikz%
1473 }

```

(End of definition for \hobby@midcurveto@auto. This function is documented on page ??.)

\hobby@finish

```

1474 \def\hobby@finish{%
1475   \hobby@restore@tikz@commands
1476   \ifhobby@externalise
1477     \ifx\hobby@path@name\pgfutil@empty
1478       \hobbygenusepath
1479     \else
1480       \hobbygenuseifnecpath{\hobby@path@name}%
1481     \fi
1482   \else
1483     \hobbygenusepath
1484   \fi

```

```

1485 \ifx\hobby@path@name\pgfutil@empty
1486 \else
1487 \hobbysavepath{\hobby@path@name}%
1488 \fi
1489 \global\let\hobby@path@name=\pgfutil@empty
1490 \tikzset{designated Hobby path=next}%
1491 }

```

(End of definition for \hobby@finish. This function is documented on page ??.)

quick_curve_through

The quick curve through is a to path which does the “quick” version of Hobby’s algorithm. The syntax is as with the curve through: to pass the midpoints as the argument to the style. We need to pass three points to the auxiliary macro. These are passed as \hobby@qpoints, \hobby@qpointa, and the current point. Then these get cycled round for the next triple. The path gets built up and stored as \hobby@quick@path. We also have to remember the angle computed for the next round.

```

1492 \tikzset{
1493   quick curve through/.style={%
1494     to path={%
1495       \pgfextra{%

```

Scan the starting point and store the coordinates in \hobby@qpointa

```

1496       \let\hobby@next@qbreak=\relax
1497       \let\hobby@next@qblank=\relax
1498       \tikz@scan@one@point\pgfutil@firstofone(\tikztostart)%
1499       \tikz@make@last@position{\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1500       \edef\hobby@qpoints{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%

```

Blank the path and auxiliary macros.

```

1501       \def\hobby@qpointa{}%
1502       \def\hobby@quick@path{}%
1503       \def\hobby@angle{}%
1504       \let\hobby@quick@curveto=\hobby@quick@makepath

```

Now start parsing the rest of the coordinates.

```

1505       \tikz@scan@one@point\hobby@quickfirst #1 (\tikztotarget)\relax
1506     }

```

Invoke the path

```

1507       \hobby@quick@path
1508     }
1509   },
1510   quick hobby/blank curve/.is choice,
1511   quick hobby/blank curve/true/.code={%
1512     \gdef\hobby@next@qblank{%
1513       \qhobby@blanktrue
1514       \global\let\hobby@next@qblank=\relax
1515     }%
1516   },
1517   quick hobby/blank curve/false/.code={%
1518     \gdef\hobby@next@qblank{%
1519       \qhobby@blankfalse
1520       \global\let\hobby@next@qblank=\relax
1521     }%
1522   },
1523   quick hobby/blank curve/once/.code={%
1524     \gdef\hobby@next@qblank{%
1525       \qhobby@blanktrue
1526       \gdef\hobby@next@qblank{%
1527         \qhobby@blankfalse
1528         \global\let\hobby@next@qblank=\relax
1529       }%

```

```

1530 }%
1531 },
1532 quick hobby/blank curve/.default=true,
1533 quick hobby/break curve/.is choice,
1534 quick hobby/break curve/true/.code={%
1535   \gdef\hobby@next@qbreak{%
1536     \qhobby@breaktrue
1537     \global\let\hobby@next@qbreak=\relax
1538   }%
1539 },
1540 quick hobby/break curve/false/.code={%
1541   \gdef\hobby@next@qbreak{%
1542     \qhobby@breakfalse
1543     \global\let\hobby@next@qbreak=\relax
1544   }%
1545 },
1546 quick hobby/break curve/once/.code={%
1547   \gdef\hobby@next@qbreak{%
1548     \qhobby@breaktrue
1549     \gdef\hobby@next@qbreak{%
1550       \qhobby@breakfalse
1551       \global\let\hobby@next@qbreak=\relax
1552     }%
1553   }%
1554 },
1555 quick hobby/break curve/.default=true,
1556 }
1557 \newif\ifqhobby@break
1558 \newif\ifqhobby@blank

```

(End of definition for quick curve through. This function is documented on page ??.)

Add plot handlers

```

1559 \tikzoption{hobby}[]{\let\tikz@plot@handler=\pgfplotthandlerhobby}
1560 \tikzoption{quick hobby}[]{\let\tikz@plot@handler=\pgfplotthandlerquickhobby}
1561 \tikzoption{closed hobby}[]{\let\tikz@plot@handler=\pgfplotthandlerclosedhobby}

```

`\hobby@quickfirst` The first time around we just set the next point.

```

1562 \def\hobby@quickfirst#1{%
1563   #1%
1564   \xdef\hobby@qpinta{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1565   \tikz@make@last@position{\hobby@qpinta}%

```

Now a check to ensure that we have more points.

```

1566   \pgfutil@ifnextchar\relax{%

```

Oops, no more points. That's not good. Bail-out.

```

1567     \xdef\hobby@quick@path{ -- (\the\pgf@x,\the\pgf@y)}%
1568   }{%

```

Okay, have more points. Phew. Call the next round. If we have dots, swallow them.

```

1569     \pgfutil@ifnextchar.{%
1570       \hobby@qswallowdots}{%
1571       \tikz@scan@one@point\hobby@quick}}

```

(End of definition for \hobby@quickfirst. This function is documented on page ??.)

`\hobby@qswallowdots` Remove dots from the input stream.

```

1572 \def\hobby@qswallowdots.{%
1573   \pgfutil@ifnextchar.{%
1574     \hobby@qswallowdots}{%
1575     \tikz@scan@one@point\hobby@quick}}

```

(End of definition for \hobby@qswallowdots. This function is documented on page ??.)

`\hobby@quick` This is our wrapper function that handles the loop.

```
1576 \def\hobby@quick#1{%
1577   \hobby@quick@compute{#1}%
1578   \tikz@make@last@position{\hobby@qpointa}%
1579   \pgfutil@ifnextchar\relax{%
```

End of loop

```
1580   \hobby@quick@computeend%
1581 }{%
```

More to go, scan in the next coordinate and off we go again.

```
1582   \pgfutil@ifnextchar.{%
1583     \hobby@qswallowdots}{%
1584       \tikz@scan@one@point\hobby@quick}}}
```

(End of definition for \hobby@quick. This function is documented on page ??.)

`\hobby@quick@makepath` Path constructor for to path use.

```
1585 \def\hobby@quick@makepath#1#2#3{%
1586   #1%
1587   \pgf@xa=\pgf@x\relax
1588   \pgf@ya=\pgf@y\relax
1589   #2%
1590   \pgf@xb=\pgf@x\relax
1591   \pgf@yb=\pgf@y\relax
1592   #3%
1593   \ifqhobby@blank
1594   \xdef\hobby@quick@path{\hobby@quick@path (\the\pgf@x,\the\pgf@y)}%
1595   \else
1596   \xdef\hobby@quick@path{\hobby@quick@path .. controls%
1597     (\the\pgf@xa,\the\pgf@ya) and (\the\pgf@xb,\the\pgf@yb) .. (\the\pgf@x,\the\pgf@y) }%
1598   \fi
1599   \ifqhobby@break
1600   \xdef\hobby@quick@path{\hobby@quick@path +(0,0)}%
1601   \fi
1602   \hobby@next@qbreak
1603   \hobby@next@qblank
1604 }
```

(End of definition for \hobby@quick@makepath. This function is documented on page ??.)

`\hobby@qcurveto@auto` Uses the “quick” method for the shortcut syntax.

```
1605 \def\hobby@qcurveto@auto{%
1606   \global\let\hobby@next@qbreak=\relax
1607   \global\let\hobby@next@qblank=\relax
1608   \xdef\hobby@qpoints{\noexpand\pgfqpoint{\the\tikz@lastx}{\the\tikz@lasty}}%
1609   \gdef\hobby@qpointa{%
1610     \gdef\hobby@quick@path{}%
1611     \gdef\hobby@angle{}%
1612     \global\let\hobby@quick@curveto=\hobby@quick@makepathauto
1613     \hobby@qinit\tikz@commands
1614     \let\tikz@scan@point@options=\pgfutil@empty
1615     \global\let\hobby@action=\pgfutil@empty
1616     \global\let\hobby@point@options=\pgfutil@empty
1617     \tikz@scan@one@point\hobby@qfirst@auto}
```

(End of definition for \hobby@qcurveto@auto. This function is documented on page ??.)

`\hobby@qmidcurveto@auto`

```
1618 \def\hobby@qmidcurveto@auto{%
1619   \let\tikz@scan@point@options=\pgfutil@empty
1620   \global\let\hobby@action=\pgfutil@empty
1621   \global\let\hobby@point@options=\pgfutil@empty
1622   \tikz@scan@one@point\hobby@qaddfromtikz}
```

(End of definition for \hobby@qmidcurveto@auto. This function is documented on page ??.)

\hobby@qfirst@auto

```
1623 \def\hobby@qfirst@auto#1{%
1624   #1%
1625   \xdef\hobby@qpointa{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1626   \tikz@make@last@position{\hobby@qpointa}%
1627   \tikz@scan@next@command%
1628 }
```

(End of definition for \hobby@qfirst@auto. This function is documented on page ??.)

\hobby@quick@makepathauto Path constructor for shortcut method to use.

```
1629 \def\hobby@quick@makepathauto#1#2#3{%
1630   #1%
1631   \pgf@xa=\pgf@x\relax
1632   \pgf@ya=\pgf@y\relax
1633   #2%
1634   \pgf@xb=\pgf@x\relax
1635   \pgf@yb=\pgf@y\relax
1636   #3%
1637   \ifqhobby@blank
1638   \edef\hobby@temp{%
1639     \noexpand\pgfpathmoveto{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1640   }%
1641   \hobby@temp
1642   \else
1643   \edef\hobby@temp{%
1644     \noexpand\pgfpathcurveto{\noexpand\pgfqpoint{\the\pgf@xa}{\the\pgf@ya}}%
1645     {\noexpand\pgfqpoint{\the\pgf@xb}{\the\pgf@yb}}%
1646     {\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1647   }%
1648   \hobby@temp
1649   \fi
1650   \ifqhobby@break
1651   #3%
1652   \edef\hobby@temp{%
1653     \noexpand\pgfpathmoveto{\noexpand\pgfqpoint{\the\pgf@x}{\the\pgf@y}}%
1654   }%
1655   \hobby@temp
1656   \fi
1657   \hobby@next@qbreak
1658   \hobby@next@qblank
1659 }
```

(End of definition for \hobby@quick@makepathauto. This function is documented on page ??.)

\hobby@qaddfromtikz This adds our current point to the stack.

```
1660 \def\hobby@qaddfromtikz#1{%
1661   \hobby@quick@compute{#1}%
1662   \tikz@make@last@position{\hobby@qpointa}%
1663   \tikz@scan@next@command%
1664 }
```

(End of definition for \hobby@qaddfromtikz. This function is documented on page ??.)

\hobby@qinit@tikz@commands

```
1665 \def\hobby@qinit@tikz@commands{%
1666   \hobby@qinit@tikz@modcmd\tikz@movetoabs
1667   \hobby@qinit@tikz@modcmd\tikz@movetorel
1668   \hobby@qinit@tikz@modcmd\tikz@lineto
1669   \hobby@qinit@tikz@modcmd\tikz@rect
```

```

1670 \hobby@qinit@tikz@modcmd\tikz@cchar
1671 \hobby@qinit@tikz@modcmd\tikz@finish
1672 \hobby@qinit@tikz@modcmd\tikz@arcA
1673 \hobby@qinit@tikz@modcmd\tikz@e@char
1674 \hobby@qinit@tikz@modcmd\tikz@g@char
1675 \hobby@qinit@tikz@modcmd\tikz@schar
1676 \hobby@qinit@tikz@modcmd\tikz@vh@lineto
1677 \hobby@qinit@tikz@modcmd\tikz@pchar
1678 \hobby@qinit@tikz@modcmd\tikz@to
1679 \hobby@qinit@tikz@modcmd\pgf@stop
1680 \hobby@qinit@tikz@modcmd\tikz@decoration
1681 \hobby@qinit@tikz@modcmd\tikz@@close
1682 \global\let\hobby@curveto@delegate=\hobby@qmidcurveto@auto
1683 }

```

(End of definition for \hobby@qinit@tikz@commands. This function is documented on page ??.)

\hobby@qrestore@tikz@commands

```

1684 \def\hobby@qrestore@tikz@commands{%
1685   \hobby@restore@tikz@modcmd\tikz@movetoabs
1686   \hobby@restore@tikz@modcmd\tikz@movetorel
1687   \hobby@restore@tikz@modcmd\tikz@lineto
1688   \hobby@restore@tikz@modcmd\tikz@rect
1689   \hobby@restore@tikz@modcmd\tikz@cchar
1690   \hobby@restore@tikz@modcmd\tikz@finish
1691   \hobby@restore@tikz@modcmd\tikz@arcA
1692   \hobby@restore@tikz@modcmd\tikz@e@char
1693   \hobby@restore@tikz@modcmd\tikz@g@char
1694   \hobby@restore@tikz@modcmd\tikz@schar
1695   \hobby@restore@tikz@modcmd\tikz@vh@lineto
1696   \hobby@restore@tikz@modcmd\tikz@pchar
1697   \hobby@restore@tikz@modcmd\tikz@to
1698   \hobby@restore@tikz@modcmd\pgf@stop
1699   \hobby@restore@tikz@modcmd\tikz@decoration
1700   \hobby@restore@tikz@modcmd\tikz@@close
1701   \global\let\hobby@curveto@delegate=\hobby@qcurveto@auto
1702 }

```

(End of definition for \hobby@qrestore@tikz@commands. This function is documented on page ??.)

\hobby@qinit@tikz@modcmd

```

1703 \def\hobby@qinit@tikz@modcmd#1{%
1704   \expandafter\global\expandafter\let\csname hobby@orig@\string#1\endcsname=#1%
1705   \gdef#1{\hobby@qfinish#1}%
1706 }

```

(End of definition for \hobby@qinit@tikz@modcmd. This function is documented on page ??.)

\hobby@qfinish

```

1707 \def\hobby@qfinish{%
1708   \hobby@quick@computeend
1709   \hobby@qrestore@tikz@commands
1710 }

```

(End of definition for \hobby@qfinish. This function is documented on page ??.)

1.4 Arrays

A lot of our data structures are really arrays. These are implemented as L^AT_EX3 “property lists”. For ease of use, an array is a property list with numeric entries together with entries “base” and “top” which hold the lowest and highest indices that have been set.

```
1711 \RequirePackage{expl3}
1712 \ExplSyntaxOn
```

Some auxiliary variables.

```
1713 \tl_new:N \l_array_tmp_tl
1714 \tl_new:N \l_array_show_tl
1715 \int_new:N \l_array_base_int
1716 \int_new:N \l_array_top_int
1717 \int_new:N \l_array_tmp_int
1718 \int_new:N \g_array_map_int
```

The global variable `\g_array_base_int` says what index a blank array should start with when pushed or unshifted.

```
1719 \int_new:N \g_array_base_int
1720 \int_gset:Nn \g_array_base_int {0}
```

`\array_adjust_ends:Nn` This ensures that the “base” and “top” are big enough to include the given index.

```
1721 \cs_new:Npn \array_adjust_ends:Nn #1#2 {
1722   \prop_get:NnNTF #1 {base} \l_tmpa_tl
1723   {
1724     \int_compare:nNnTF {\l_tmpa_tl} > {#2}
1725     {
1726       \prop_put:Nnx #1 {base} {\int_eval:n {#2}}
1727     }
1728     {}
1729   }
1730   {
1731     \prop_put:Nnx #1 {base} {\int_eval:n {#2}}
1732   }
1733   \prop_get:NnNTF #1 {top} \l_tmpa_tl
1734   {
1735     \int_compare:nNnTF {\l_tmpa_tl} < {#2}
1736     {
1737       \prop_put:Nnx #1 {top} {\int_eval:n {#2}}
1738     }
1739     {}
1740   }
1741   {
1742     \prop_put:Nnx #1 {top} {\int_eval:n {#2}}
1743   }
1744 }
```

(End of definition for `\array_adjust_ends:Nn`. This function is documented on page ??.)

`\array_gadjust_ends:Nn` This ensures that the “base” and “top” are big enough to include the given index. (Global version)

```
1745 \cs_new:Npn \array_gadjust_ends:Nn #1#2 {
1746   \prop_get:NnNTF #1 {base} \l_tmpa_tl
1747   {
1748     \int_compare:nNnTF {\l_tmpa_tl} > {#2}
1749     {
1750       \prop_gput:Nnx #1 {base} {\int_eval:n {#2}}
1751     }
1752     {}
1753   }
1754   {
1755     \prop_gput:Nnx #1 {base} {\int_eval:n {#2}}
```

```

1756 }
1757 \prop_get:NnNTF #1 {top} \l_tmpa_tl
1758 {
1759   \int_compare:nNnTF {\l_tmpa_tl} < {#2}
1760   {
1761     \prop_gput:Nnx #1 {top} {\int_eval:n {#2}}
1762   }
1763   {}
1764 }
1765 {
1766   \prop_gput:Nnx #1 {top} {\int_eval:n {#2}}
1767 }
1768 }

```

(End of definition for \array_gadjust_ends:Nn. This function is documented on page ??.)

\array_put:Nnn When adding a value to an array we have to adjust the ends.

```

1769 \cs_new:Npn \array_put:Nnn #1#2#3 {
1770   \exp_args:NNx \prop_put:Nnn #1 {\int_eval:n {#2}} {#3}
1771   \array_adjust_ends:Nn #1{#2}
1772 }
1773 \cs_generate_variant:Nn \array_put:Nnn {Nnx}

```

(End of definition for \array_put:Nnn. This function is documented on page ??.)

\array_gput:Nnn When adding a value to an array we have to adjust the ends. (Global version)

```

1774 \cs_new:Npn \array_gput:Nnn #1#2#3 {
1775   \exp_args:NNx \prop_gput:Nnn #1 {\int_eval:n {#2}} {#3}
1776   \array_gadjust_ends:Nn #1{#2}
1777 }
1778 \cs_generate_variant:Nn \array_gput:Nnn {Nnx}

```

(End of definition for \array_gput:Nnn. This function is documented on page ??.)

\array_get:NnN

```

1779 \cs_new:Npn \array_get:NnN #1#2#3 {
1780   \exp_args:NNx \prop_get:NnN #1 {\int_eval:n {#2}} #3
1781 }

```

(End of definition for \array_get:NnN. This function is documented on page ??.)

\array_get:Nn

```

1782 \cs_new:Npn \array_get:Nn #1#2 {
1783   \exp_args:Nnf \prop_item:Nn #1 { \int_eval:n {#2} }
1784 }

```

(End of definition for \array_get:Nn. This function is documented on page ??.)

\array_get:NnNTF

```

1785 \cs_new:Npn \array_get:NnNTF #1#2#3#4#5 {
1786   \exp_args:NNx \prop_get:NnNTF #1 {\int_eval:n {#2}} #3 {#4}{#5}
1787 }

```

(End of definition for \array_get:NnNTF. This function is documented on page ??.)

\array_if_empty:NTF

```

1788 \prg_new_conditional:Npnn \array_if_empty:N #1 { p, T, F, TF }
1789 {
1790   \if_meaning:w #1 \c_empty_prop
1791   \prg_return_true:
1792   \else:
1793     \prg_return_false:
1794   \fi:
1795 }

```

(End of definition for \array_if_empty:NTF. This function is documented on page ??.)

\array_if_exist:NTF

```
1796 \prg_new_eq_conditional:NnN \array_if_exist:N \cs_if_exist:N { p, T, F, TF }
```

(End of definition for \array_if_exist:NTF. This function is documented on page ??.)

\array_new:N

```
1797 \cs_new_eq:NN \array_new:N \prop_new:N
```

(End of definition for \array_new:N. This function is documented on page ??.)

\array_clear:N

```
1798 \cs_new_eq:NN \array_clear:N \prop_clear:N
```

(End of definition for \array_clear:N. This function is documented on page ??.)

\array_gclear:N

```
1799 \cs_new_eq:NN \array_gclear:N \prop_gclear:N
```

(End of definition for \array_gclear:N. This function is documented on page ??.)

\array_map_function

When stepping through an array, we want to iterate in order so a simple wrapper to \prop_map_function is not enough. This maps through every value from the base to the top so the function should be prepared to deal with a \q_no_value.

```
1800 \cs_new:Npn \array_map_function:NN #1#2
1801 {
1802   \array_if_empty:NTF #1 {} {
1803     \prop_get:NnNTF #1 {base} \l_array_tmp_tl {
1804       \int_set:Nn \l_array_base_int {\l_array_tmp_tl}
1805     }{
1806       \int_set:Nn \l_array_base_int {0}
1807     }
1808     \prop_get:NnNTF #1 {top} \l_array_tmp_tl {
1809       \int_set:Nn \l_array_top_int {\l_array_tmp_tl}
1810     }{
1811       \int_set:Nn \l_array_top_int {0}
1812     }
1813     \int_step_inline:nnnn {\l_array_base_int} {1} {\l_array_top_int} {
1814       \array_get:NnN #1 {##1} \l_array_tmp_tl
1815       \exp_args:NnV #2 {##1} \l_array_tmp_tl
1816     }
1817   } {}
1818 }
1819 \cs_generate_variant:Nn \array_map_function:NN { Nc }
1820 \cs_generate_variant:Nn \array_map_function:NN { c , cc }
```

(End of definition for \array_map_function. This function is documented on page ??.)

\array_reverse_map_function

This steps through the array in reverse order.

```
1821 \cs_new:Npn \array_reverse_map_function:NN #1#2
1822 {
1823   \array_if_empty:NTF #1 {} {
1824     \prop_get:NnNTF #1 {base} \l_array_tmp_tl {
1825       \int_set:Nn \l_array_base_int {\l_array_tmp_tl}
1826     }{
1827       \int_set:Nn \l_array_base_int {0}
1828     }
1829     \prop_get:NnNTF #1 {top} \l_array_tmp_tl {
1830       \int_set:Nn \l_array_top_int {\l_array_tmp_tl}
1831     }{
1832       \int_set:Nn \l_array_top_int {0}
```

```

1833     }
1834     \int_step_inline:nnnn {\l_array_top_int} {-1} {\l_array_base_int} {
1835     \array_get:NnN #1 {##1} \l_array_tmp_tl
1836     \exp_args:Nno #2 {##1} \l_array_tmp_tl
1837   }
1838 } {}
1839 }
1840 \cs_generate_variant:Nn \array_reverse_map_function:NN { Nc }
1841 \cs_generate_variant:Nn \array_reverse_map_function:NN { c , cc }

```

(End of definition for \array_reverse_map_function. This function is documented on page ??.)

\array_map_inline:Nn Inline version of the above.

```

1842 \cs_new_protected:Npn \array_map_inline:Nn #1#2
1843 {
1844   \int_gincr:N \g_array_map_int
1845   \cs_gset:cpn { array_map_inline_ \int_use:N \g_array_map_int :nn }
1846   ##1##2 {#2}
1847   \exp_args:NNc \array_map_function:NN #1
1848   { array_map_inline_ \int_use:N \g_array_map_int :nn }
1849   \prg_break_point:Nn \array_map_break: { \int_gdecr:N \g_array_map_int }
1850 }
1851 \cs_generate_variant:Nn \array_map_inline:Nn { c }

```

(End of definition for \array_map_inline:Nn. This function is documented on page ??.)

\array_reverse_map_inline:Nn Inline version of the above.

```

1852 \cs_new_protected:Npn \array_reverse_map_inline:Nn #1#2
1853 {
1854   \int_gincr:N \g_array_map_int
1855   \cs_gset:cpn { array_map_inline_ \int_use:N \g_array_map_int :nn }
1856   ##1##2 {#2}
1857   \exp_args:NNc \array_reverse_map_function:NN #1
1858   { array_map_inline_ \int_use:N \g_array_map_int :nn }
1859   \prg_break_point:Nn \array_map_break: { \int_gdecr:N \g_array_map_int }
1860 }
1861 \cs_generate_variant:Nn \array_reverse_map_inline:Nn { c }

```

(End of definition for \array_reverse_map_inline:Nn. This function is documented on page ??.)

\array_map_break:

```

1862 \cs_new_nopar:Npn \array_map_break:
1863 { \prg_map_break:Nn \array_map_break: { } }
1864 \cs_new_nopar:Npn \array_map_break:n
1865 { \prg_map_break:Nn \array_map_break: }

```

(End of definition for \array_map_break:. This function is documented on page ??.)

For displaying arrays, we need some messages.

```

1866 \msg_new:nnn { kernel } { show-array }
1867 {
1868   The~array~\token_to_str:N #1~
1869   \array_if_empty:NTF #1
1870   { is~empty }
1871   { contains~the~items~(without~outer~braces): }
1872 }

```

\array_show:N Mapping through an array isn't expandable so we have to set a token list to its contents first before passing it to the message handler.

```

1873 \cs_new_protected:Npn \array_show:N #1
1874 {
1875   \__msg_show_variable:NNNnn

```

```

1876     #1
1877     \array_if_exist:NTF
1878     \array_if_empty:NTF
1879     { array }
1880     { \array_map_function:NN #1 \_msg_show_item:nn }
1881   }
1882 \cs_generate_variant:Nn \array_show:N { c }

```

(End of definition for \array_show:N. This function is documented on page ??.)

\array_push:Nn

```

1883 \cs_new_protected:Npn \array_push:Nn #1#2
1884 {
1885   \prop_get:NnNTF #1 {top} \l_array_tmp_tl
1886   {
1887     \int_set:Nn \l_array_tmp_int {\l_array_tmp_tl}
1888     \int_incr:N \l_array_tmp_int
1889     \array_put:Nnn #1 {\l_array_tmp_int} {#2}
1890   }
1891   {
1892     \array_put:Nnn #1 {\g_array_base_int} {#2}
1893   }
1894 }
1895 \cs_generate_variant:Nn \array_push:Nn {Nx}

```

(End of definition for \array_push:Nn. This function is documented on page ??.)

\array_gpush:Nn

```

1896 \cs_new_protected:Npn \array_gpush:Nn #1#2
1897 {
1898   \prop_get:NnNTF #1 {top} \l_array_tmp_tl
1899   {
1900     \int_set:Nn \l_array_tmp_int {\l_array_tmp_tl}
1901     \int_incr:N \l_array_tmp_int
1902     \array_gput:Nnn #1 {\l_array_tmp_int} {#2}
1903   }
1904   {
1905     \array_gput:Nnn #1 {\g_array_base_int} {#2}
1906   }
1907 }
1908 \cs_generate_variant:Nn \array_gpush:Nn {Nx}

```

(End of definition for \array_gpush:Nn. This function is documented on page ??.)

\array_unshift:Nn

```

1909 \cs_new_protected:Npn \array_unshift:Nn #1#2
1910 {
1911   \prop_get:NnNTF #1 {base} \l_array_tmp_tl
1912   {
1913     \int_set:Nn \l_array_tmp_int {\l_array_tmp_tl}
1914     \int_decr:N \l_array_tmp_int
1915     \array_put:Nnn #1 {\l_array_tmp_int} {#2}
1916   }
1917   {
1918     \array_put:Nnn #1 {\g_array_base_int} {#2}
1919   }
1920 }
1921 \cs_generate_variant:Nn \array_unshift:Nn {Nx}

```

(End of definition for \array_unshift:Nn. This function is documented on page ??.)

\array_gunshift:Nn

```
1922 \cs_new_protected:Npn \array_gunshift:Nn #1#2
1923 {
1924   \prop_get:NnNTF #1 {base} \l_array_tmp_tl
1925   {
1926     \int_set:Nn \l_array_tmp_int {\l_array_tmp_tl}
1927     \int_decr:N \l_array_tmp_int
1928     \array_gput:Nnn #1 {\l_array_tmp_int} {#2}
1929   }
1930   {
1931     \array_gput:Nnn #1 {\g_array_base_int} {#2}
1932   }
1933 }
1934 \cs_generate_variant:Nn \array_gunshift:Nn {Nx}
```

(End of definition for \array_gunshift:Nn. This function is documented on page ??.)

\array_pop:NN

```
1935 \cs_new_protected:Npn \array_pop:NN #1#2
1936 {
1937   \prop_get:NnN #1 {top} \l_array_tmp_tl
1938   \array_get:NnN #1 {\l_array_tmp_tl} #2
1939   \array_del:Nn #1 {\l_array_tmp_tl}
1940 }
```

(End of definition for \array_pop:NN. This function is documented on page ??.)

\array_gpop:NN

```
1941 \cs_new_protected:Npn \array_gpop:NN #1#2
1942 {
1943   \prop_get:NnN #1 {top} \l_array_tmp_tl
1944   \array_get:NnN #1 {\l_array_tmp_tl} #2
1945   \array_gdel:Nn #1 {\l_array_tmp_tl}
1946 }
```

(End of definition for \array_gpop:NN. This function is documented on page ??.)

\array_shift:NN

```
1947 \cs_new_protected:Npn \array_shift:NN #1#2
1948 {
1949   \prop_get:NnN #1 {base} \l_array_tmp_tl
1950   \array_get:NnN #1 {\l_array_tmp_tl} #2
1951   \array_del:Nn #1 {\l_array_tmp_tl}
1952 }
```

(End of definition for \array_shift:NN. This function is documented on page ??.)

\array_gshift:NN

```
1953 \cs_new_protected:Npn \array_gshift:NN #1#2
1954 {
1955   \prop_get:NnN #1 {base} \l_array_tmp_tl
1956   \array_get:NnN #1 {\l_array_tmp_tl} #2
1957   \array_gdel:Nn #1 {\l_array_tmp_tl}
1958 }
```

(End of definition for \array_gshift:NN. This function is documented on page ??.)

\array_top:NN

```
1959 \cs_new_protected:Npn \array_top:NN #1#2
1960 {
1961   \prop_get:NnN #1 {top} \l_array_tmp_tl
1962   \array_get:NnN #1 {\l_array_tmp_tl} #2
1963 }
```

(End of definition for \array_top:NN. This function is documented on page ??.)

\array_base:NN

```

1964 \cs_new_protected:Npn \array_base:NN #1#2
1965 {
1966   \prop_get:NnN #1 {base} \l_array_tmp_tl
1967   \array_get:NnN #1 {\l_array_tmp_tl} #2
1968 }

```

(End of definition for \array_base:NN. This function is documented on page ??.)

\array_top:N

```

1969 \cs_new:Npn \array_top:N #1
1970 {
1971   \array_get:Nn #1 {\prop_item:Nn #1 {top}}
1972 }

```

(End of definition for \array_top:N. This function is documented on page ??.)

\array_base:N

```

1973 \cs_new:Npn \array_base:N #1
1974 {
1975   \array_get:Nn #1 {\prop_item:Nn #1 {base}}
1976 }

```

(End of definition for \array_base:N. This function is documented on page ??.)

\array_del:Nn

```

1977 \cs_new_protected:Npn \array_del:Nn #1#2
1978 {
1979   \exp_args:NNx \prop_pop:Nn #1 {\int_eval:n {#2}}
1980   \int_set:Nn \l_array_tmp_int {0}
1981   \array_map_inline:Nn #1 {
1982     \tl_if_eq:NNTF {##2} {\q_no_value} {}
1983     {
1984       \int_incr:N \l_array_tmp_int
1985     }
1986   }
1987   \int_compare:nNnTF {\l_array_tmp_int} = {0}
1988   {
1989     \prop_clear:N #1
1990   }
1991   {
1992     \prop_get:NnN #1 {top} \l_array_tmp_tl
1993     \int_compare:nNnTF {#2} = {\l_array_tmp_tl} {
1994       \prop_get:NnN #1 {base} \l_array_tmp_tl
1995       \int_set:Nn \l_array_tmp_int {\l_array_tmp_tl}
1996       \array_map_inline:Nn #1 {
1997         \tl_if_eq:NNTF {##2} {\q_no_value} {}
1998         {
1999           \int_compare:nNnTF {\l_array_tmp_int} < {##1} {
2000             \int_set:Nn \l_array_tmp_int {##1}
2001           }{}
2002         }
2003       }
2004       \prop_put:Nnx #1 {top} {\int_use:N \l_array_tmp_int}
2005     }{}
2006     \prop_get:NnN #1 {base} \l_array_tmp_tl
2007     \int_compare:nNnTF {#2} = {\l_array_tmp_tl} {
2008       \prop_get:NnN #1 {top} \l_array_tmp_tl
2009       \int_set:Nn \l_array_tmp_int {\l_array_tmp_tl}
2010       \array_map_inline:Nn #1 {

```

```

2011     \tl_if_eq:NNTF {##2} {\q_no_value} {}
2012     {
2013         \int_compare:nNnTF {\l_array_tmp_int} > {##1} {
2014             \int_set:Nn \l_array_tmp_int {##1}
2015         }{}
2016     }
2017 }
2018 \prop_put:Nnx #1 {base} {\int_use:N \l_array_tmp_int}
2019 }{}
2020 }
2021 }

```

(End of definition for \array_del:Nn. This function is documented on page ??.)

\array_gdel:Nn

```

2022 \cs_new_protected:Npn \array_gdel:Nn #1#2
2023 {
2024     \exp_args:NNx \prop_gremove:Nn #1 {\int_eval:n {#2}}
2025     \int_set:Nn \l_array_tmp_int {0}
2026     \array_map_inline:Nn #1 {
2027         \tl_if_eq:NNTF {##2} {\q_no_value} {}
2028         {
2029             \int_incr:N \l_array_tmp_int
2030         }
2031     }
2032     \int_compare:nNnTF {\l_array_tmp_int} = {0}
2033     {
2034         \prop_gclear:N #1
2035     }
2036     {
2037         \prop_get:NnN #1 {top} \l_array_tmp_tl
2038         \int_compare:nNnTF {#2} = {\l_array_tmp_tl} {
2039             \prop_get:NnN #1 {base} \l_array_tmp_tl
2040             \int_set:Nn \l_array_tmp_int {\l_array_tmp_tl}
2041             \array_map_inline:Nn #1 {
2042                 \tl_if_eq:NNTF {##2} {\q_no_value} {}
2043                 {
2044                     \int_compare:nNnTF {\l_array_tmp_int} < {##1} {
2045                         \int_set:Nn \l_array_tmp_int {##1}
2046                     }{}
2047                 }
2048             }
2049             \prop_gput:Nnx #1 {top} {\int_use:N \l_array_tmp_int}
2050         }{}
2051         \prop_get:NnN #1 {base} \l_array_tmp_tl
2052         \int_compare:nNnTF {#2} = {\l_array_tmp_tl} {
2053             \prop_get:NnN #1 {top} \l_array_tmp_tl
2054             \int_set:Nn \l_array_tmp_int {\l_array_tmp_tl}
2055             \array_map_inline:Nn #1 {
2056                 \tl_if_eq:NNTF {##2} {\q_no_value} {}
2057                 {
2058                     \int_compare:nNnTF {\l_array_tmp_int} > {##1} {
2059                         \int_set:Nn \l_array_tmp_int {##1}
2060                     }{}
2061                 }
2062             }
2063             \prop_gput:Nnx #1 {base} {\int_use:N \l_array_tmp_int}
2064         }{}
2065     }
2066 }

```

(End of definition for \array_gdel:Nn. This function is documented on page ??.)


```

\array_length:N
2067 \cs_new_protected:Npn \array_length:N #1
2068 {
2069   \int_eval:n {\prop_item:Nn #1 {top} - \prop_item:Nn #1 {base}}
2070 }

(End of definition for \array_length:N. This function is documented on page ??.)

2071 \ExplSyntaxOff
2072 \RequirePackage{expl3}
    Load the hobby core
2073 \input{hobby.code.tex}
    Register as an expl3 package
2074 \ProvidesExplPackage {hobby-l3draw} {2018/02/20} {1.0} {Interface for l3draw and hobby}
    Load the l3draw package
2075 \RequirePackage{l3draw}

\hobby_draw_moveto:nnn This provides our interface between hobby's moveto and l3draw's moveto
2076 \cs_new_protected:Npn \hobby_draw_moveto:nnn #1#2#3
2077 {
2078   \draw_path_canvas_moveto:n {#3}
2079 }

(End of definition for \hobby_draw_moveto:nnn. This function is documented on page ??.)

hobby_draw_curveto:nnn This provides our interface between hobby's curveto and l3draw's curveto
2080 \cs_set_eq:NN \hobby_draw_curveto:nnn \draw_path_canvas_curveto:nnn

(End of definition for \hobby_draw_curveto:nnn. This function is documented on page ??.)

\hobby_draw_close:n This provides our interface between hobby's close and l3draw's close
2081 \cs_new_protected:Npn \hobby_draw_close:n #1
2082 {
2083   \draw_path_close:
2084 }

(End of definition for \hobby_draw_close:n. This function is documented on page ??.)

\hobby_draw_addpoint:n This processes a point and passes it one more step towards the hobby algorithm
2085 \cs_new_protected:Npn \hobby_draw_addpoint:n #1
2086 {
2087   \__draw_point_process:nn
2088   { \__hobby_draw_addpoint:nn }
2089   { \draw_point_transform:n {#1} }
2090 }

(End of definition for \hobby_draw_addpoint:n. This function is documented on page ??.)

\__hobby_draw_curveto:nn This provides our interface between l3draw's points and hobby's syntax
2091 \cs_new_protected:Npn \__hobby_draw_addpoint:nn #1#2
2092 {
2093   \hobby_add_point:nnnnnn {#1} {#2} {1} {1} {0} {10}
2094 }

(End of definition for \__hobby_draw_curveto:nn.)

\hobby_draw_init: This initialises hobby's algorithm with the l3draw commands
2095 \cs_new_protected:Npn \hobby_draw_init:
2096 {
2097   \hobby_set_cmds:NNN \hobby_draw_moveto:nnn \hobby_draw_curveto:nnn \hobby_draw_close:n
2098   \hobby_clear_path:
2099 }

(End of definition for \hobby_draw_init:. This function is documented on page ??.)

```