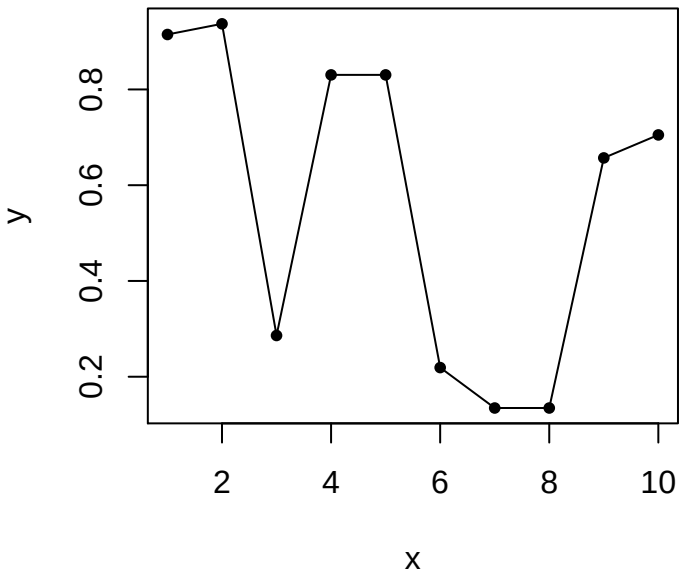
THE STALKER SPLINE

SIMEN GAURE

Abstract. The idea behind the stalker spline in package **chebpol** is outlined. It is designed as a one-dimensional interpolation to be almost shape preserving in the sense that it attempts to honour monotonocity properties and local extreme points in the data, though not entirely. There is no fancy theory behind, but it served a purpose for the author, and here it is. The near monotonicity is achieved by somewhat non-traditional means, by sacrificing analyticity, and in a corner case even differentiability. That is, the derivative is not well behaved. The name comes from the fact that it follows the data frighteningly close, though it sometimes seems stupid with little foresight. **chebpol** contains higher dimensional versions too, but they could as well be called *wet paper spline*. The interpolation isn't a proper spline, since it is not a polynomial. We also obtain a limit in intuitive geometric terms on how large the overshoot can be. The stalker spline can be used when a little more smoothness than multilinear is required.

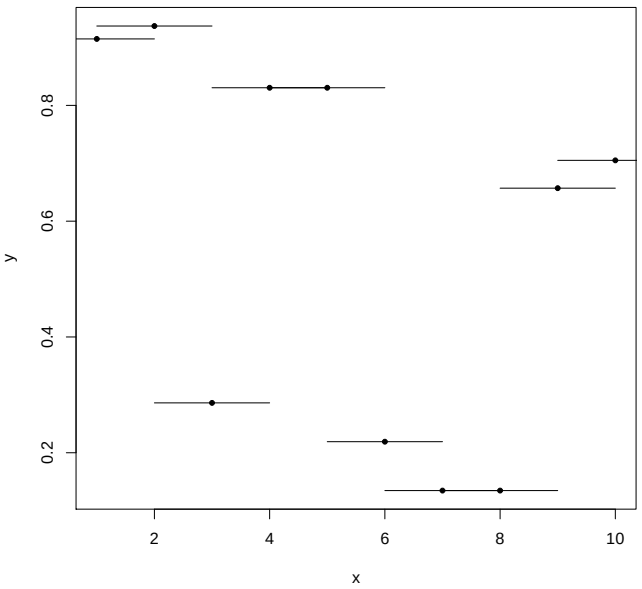
1. Introduction

The multilinear interpolation in **chebpol** is easy to understand. We have some points on the x-axis. At every point we have a value, and we just draw straight lines between the "knots":



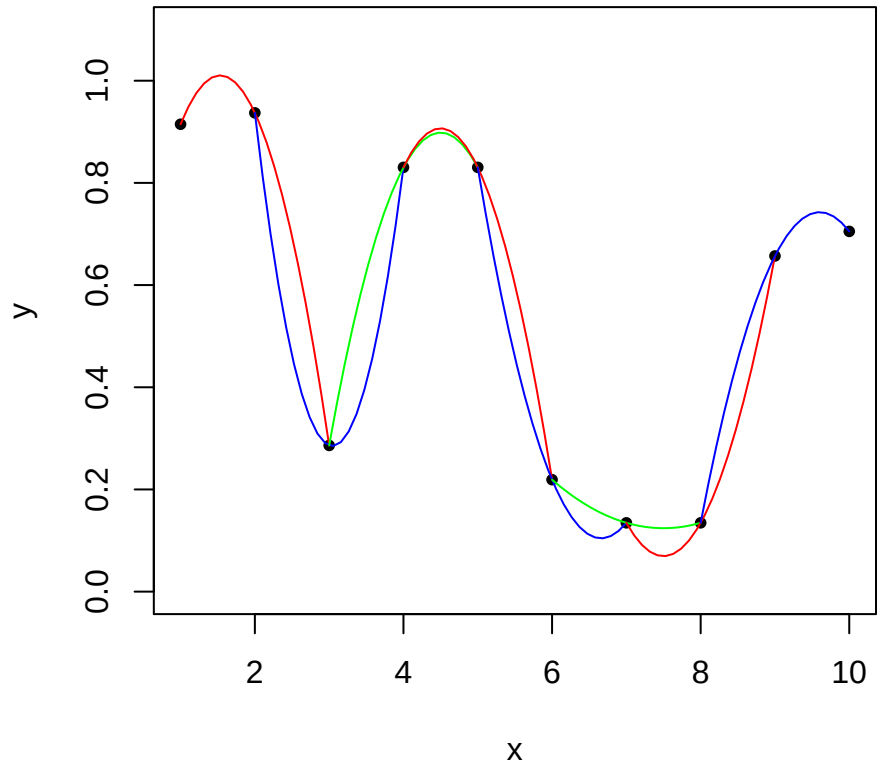
There is another way to think about this, we can imagine that at each knot i there lives a function $f_i(x)$. Whenever we are between two knots, the interpolated value is a convex combination of the two functions at each side. Say we are in $x = 3.3$. We should have a part of $f_3(3.3)$, and a part of $f_4(3.3)$, the value should be $0.7f_3(3.3) + 0.3f_4(3.3)$, or more generally for $3 \leq x \leq 4$, $tf_3(x) + (1-t)f_4(x)$ where $0 \leq t = 4 - x \leq 1$. In the multilinear case, all the functions are constant, and equal to the value in the point where it lives.

Date: July 28, 2018.



When we make a convex combination of two constant values v_i and v_{i+1}, we obtain a straight line: $tv_i + (1-t)v_{i+1}$. This linear interpolation is faithful to the data in the sense that it honours local extrema as well as monotonicity.

In the stalker spline we replace these constant functions with non-constant functions. I.e. the function $f_i(x)$ should not be constant, but pass through the 3 knots $i-1, i$ and $i+1$. A classical method is to let f_i be the unique quadratic which passes through the three knots.



The interpolated value between two knots is still a convex combination of the two functions living there. The result is that the interpolant between any two points is a cubic, with a value between the two functions living there.

In the random points we have chosen, we have deliberately made the fourth and fifth points equal. The 7th and 8th differ by only 0.00001. This accentuates a phenomenon which in some cases can be a problem, no polynomial except for the constant can be constant on an interval. There is "overshoot" between knots 4 and 5. Indeed, many of the functions overshoot, like between 7 and 8.

The idea behind the stalker spline is to reduce the overshoot, this is achieved by ensuring that if the knots $i-1, i$ and $i+1$ are monotonic (either increasing or decreasing), then the function $f_i(x)$ will also be monotonic. Other splines, like the Fritsch-Carlson spline in `splinefun(...,method='monoH.FC')` also does something similar, though with proper polynomial spline. We achieve this by non-traditionally using a fractional degree, i.e. a function of the form $a+bx+c|x|^r$, with $1 \le r \le 2$. Alternatively, we use a hyperbolic function $a+bx+\frac{d}{1+cx}$. In this note we call these functions defined by three points and a monotonicity constraint, *basis functions*. They are not basis functions in the sense used in the spline literature; they are not glued together by scalar weights, but by linear functions.

2. The stalker spline

To simplify, we consider a basis function $f(x)$ on the interval $[-1,1]$. We have its function values in the three knots $f(-1)=v_-$, $f(0)=v_0$, and $f(1)=v_+$. We assume

$$f(x) = a + bx + c|x|^r. \tag{1}$$

Inserting our three points, we obtain three equations with three unknowns:

$$\begin{aligned} a - b + c &= v_-, \\ a &= v_0, \\ a + b + c &= v_+. \end{aligned} \tag{2}$$

The solution is

$$\begin{aligned} a &= v_0, \\ b &= \frac{1}{2}(v_+ - v_-), \\ c &= \frac{1}{2}(v_+ + v_-) - v_0. \end{aligned} \tag{3}$$

These coefficients will work with any r. Typically we will pick $r=2$, but this may destroy monotonicity. The three knots are monotonic (either increasing or decreasing) whenever $|c|<|b|$. This can be seen from equation (2). Monotonicity occurs when $v_+ - v_0 = b+c$ has the same sign as $v_0 - v_- = b-c$, which is precisely when $|c|<|b|$.

If this is the case we will use monotonicity to find a suitable r. To be specific, we have

$$f'(x) = b + cr|x|^{r-1}\operatorname{sgn}(x), \tag{4}$$

where $\operatorname{sgn}(x)$ is the sign function. We have a critical point $f'(x)=0$ for $|x|^{r-1}\operatorname{sgn}(x) = -b/(cr)$. This equation has a solution in $-1<x<1$ if $|b|<r|c|$. We pick an r so that the critical point disappears from the interior. More specifically, if $r=2$ results in non-monotonicity, i.e. if $|b|<2|c|$, we pick the largest r which will make $f(x)$ monotonic. That is, $r=|b/c|<2$, this will relegate the critical point to one of the end points.

The same exercise with a non-uniform grid results in a non-linear equation in r which is solved numerically by **chebpol**.

There are some special cases. What if $c=0$? This only happens when the knots are collinear, i.e. on a straight line, but then r is irrelevant, so we do not need to compute it. What about the corner case $|b|=|c|$? This happens if v_0 equals either v_- or v_+, i.e. if we have a horizontal region. In this special case, $r=1$, and we have a non-differentiable $f(x)=a+b(x\pm|x|)$, which is constant on one side of 0 and linear on the other.

At the outset, we only need to adjust r away from 2 when there is monotonicity which is violated by a quadratic, i.e. when $|c|<|b|<2|c|$. If we stick strictly to this idea, it means that as soon as $|b|<|c|$, we will change the degree r from 1 to 2 in a jump. That is, for $|b|=|c|$ we have a constant/linear function, but if $|b|$ decreases ever so little, $|b|=|c|-\epsilon$, we suddenly shift to a quadratic which may have considerable overshoot. To make this transition smoother, we (somewhat arbitrarily) set $r=|c/b|$ whenever $|c/2|<|b|<|c|$, i.e. we gradually creep back to $r=2$.

In figure 1 is a plot of some of the functions for the case $v_-=0$, $v_+=1$, with varying v_0 (the black dots).

The dark blue curves are quadratic. The blue curves are monotonic, but with lowered degree. The light blue curves are non-monotonic with lowered degree. Every function except for the two obvious ones are continuously differentiable, but not twice differentiable in 0. If we let $|v_0|$ grow further, the function will stay quadratic, and the extreme point and the overshoot will converge to 0. Ideally, the non-monotonic curves should have an extreme point in $x=0$, i.e. no overshoot, but that is impossible with this function form.

If either v_- or v_+ grows, $|b/c|$ converges to 1, so we will converge to the non-differentiable case. However, there is a problem which becomes pressing when trying to use the stalker spline in two or more dimensions. If, as in figure 2, v_-

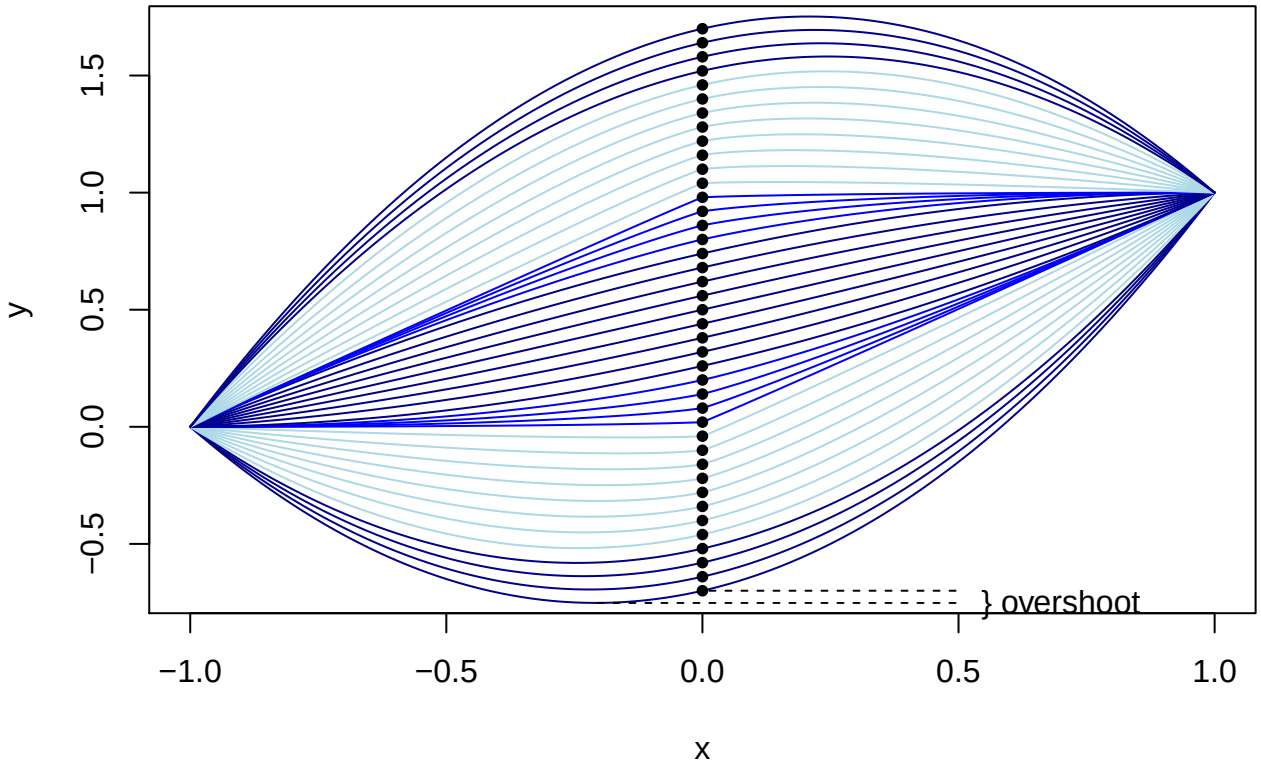


FIGURE 1. "Basis functions"

moves, the curve for $x < 0$ moves down, but the curve for $x > 0$ moves up or down depending on whether the degree changes or not. This behaviour creates problems e.g. for surfaces with torsion, as we shall see later.

The stalker spline also contains hyperbolic functions of the type $a + bx + \frac{d}{1+cx}$ which are guaranteed to honour both monotonicity and local extrema, but like the varying degree stalker it does not generalize well to higher dimensions, more or less because "monotonic" isn't such a simple concept there. Again we have four parameters and three points. For monotonic points we use monotonocity as a fourth constraint, or more precisely we let $b = 0$. For non-monotonic points, we use that the middle point should be a local extremum. Thus, the hyperbolic stalker respects both monotonicity and local extrema, but still at the cost of non-differentiability in the case of completely flat regions. In another special case, it reduces to a parabola. The hyperbolic stalker is specified in chebpol as a stalker with zero degree.

3. Overshoot and blending

To sum up, the function passing through $(-1, v_-)$, $(0, v_0)$ and $(1, v_+)$ is

$$f(x) = a + bx + c|x|^r. \tag{5}$$

The coefficients a, b, and c are as in (3). For $c \neq 0$ the exponent r is chosen as follows:

$$r = \begin{cases} |b/c| & \text{for } |c| \leq |b| < 2|c|, \\ |c/b| & \text{for } |b| < |c| < 2|b|, \\ 2 & \text{otherwise.} \end{cases} \tag{6}$$

It is clear that $1 \leq r \leq 2$. The functions with $1 < r \leq 2$ are everywhere differentiable, but for $r < 2$ the second derivative is unbounded near 0, so the graph may turn arbitrarily abruptly in the knots.

We define the overshoot as 0 for monotonic knots, and as $|v_0 - f(x_0)|$ where x_0 is the critical point of the function: $f'(x_0) = 0$. From equation (4) we have by elementary calculus $x_0 = -\operatorname{sgn}(bc)|b/(rc)|^{1/(r-1)}$. If $|b| >= 2|c|$ then

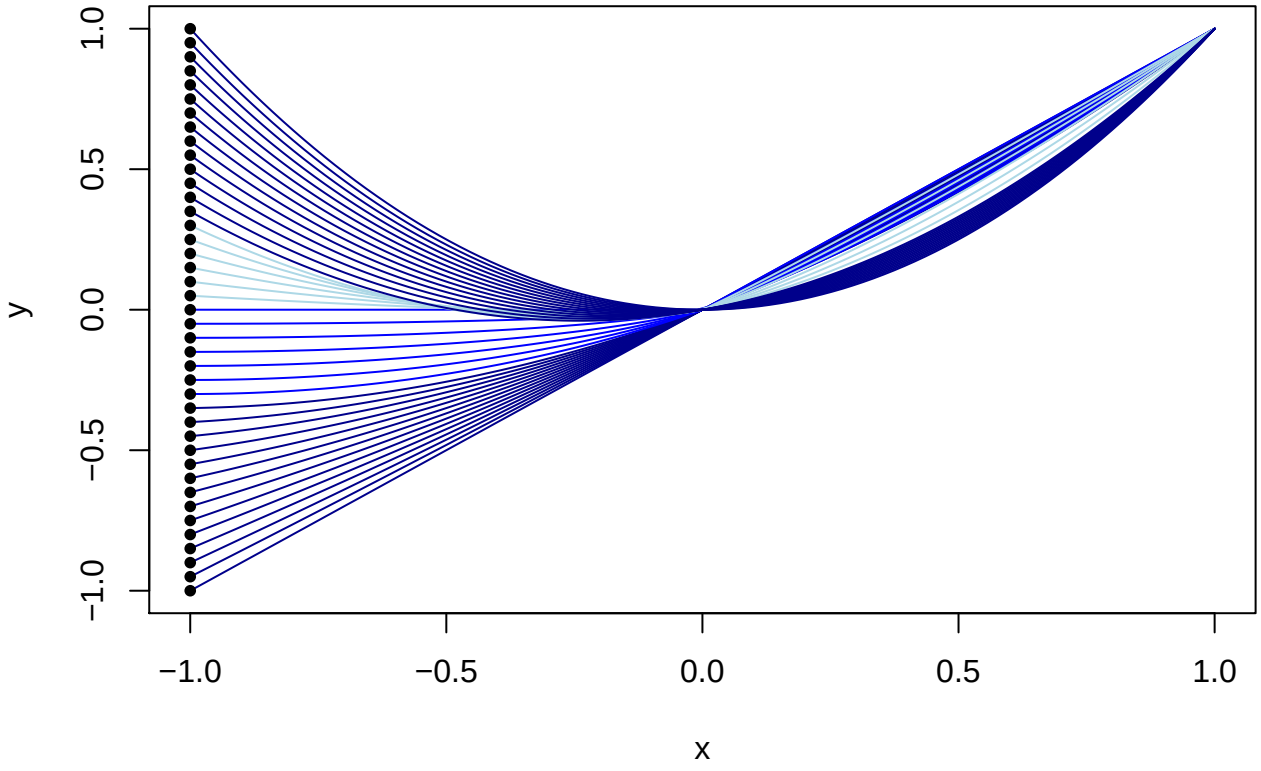


FIGURE 2. "Basis functions"

$x_0 \notin (-1, 1)$, hence the min/max occurs in an end point, in which case it is not an overshoot. We look at the case $|b| < r|c|$.

For $r = 2$ we must by definition have $|c| \geq 2|b|$. We have $x_0 = -b/(2c)$. The function value in $-b/(2c)$ is $a - b^2/(4c)$, the vertical distance from the point $v_0 = a$ is $d = |b^2/(4c)| \leq |b|/8$. Now, $|b|$ is half the distance between v_+ and v_-, so the overshoot will always be smaller than $|v_+ - v_-|/16$ for $r = 2$.

For $r < 2$, r is either equal to $|b/c|$, in which case there is no overshoot, or $r = |c/b|$ when $|c/2| < |b| < |c|$. We have $x_0 = -\operatorname{sgn}(bc)|b/(rc)|^{1/(r-1)} = -\operatorname{sgn}(bc)r^{2/(1-r)}$.

The function value in the extreme point x_0 is then $a - |b|\operatorname{sgn}(c)r^{2/(1-r)} + cr^{2r/(1-r)}$. The distance to $v_0 = a$ is $d_r = \left||c|r^{2r/(1-r)} - |b|r^{2/(1-r)}\right| = |b|\left|r^{(1+r)/(1-r)} - r^{2/(1-r)}\right|$. Again, it is easy to show by elementary calculus that d_r is increasing in r, so $d_r \leq |b|/8$.

We can prove more. If we have $|b| < |c|$ it means that v_0 is the smallest or the largest of the three knots. To simplify we assume that $v_0 < v_- < v_+$, i.e. that $0 < b < c < 2b$. The situation is symmetric with the direction reversed. We have that $c - b = v_- - v_0$, the amount that the middle knot is below the next lowest knot. We also have $c - b = b(r - 1)$. If we compute the overshoot as a fraction of this "knot overshoot": $d_r/(c - b) = d_r/(b(r - 1))$, again we get $d_r/(v_- - v_0) = |r^{(1+r)/(1-r)} - r^{2/(1-r)}|/(r - 1) \leq e^{-2} \approx 0.135$.

In short, the overshoot is always less than 14% of the vertical distance from the lowest/highest knot to the next lowest/highest knot.

With $f(0) = 0$, the hyperbolic stalker is

$$f(x) = a + bx - \frac{a}{1 + cx}. \tag{7}$$

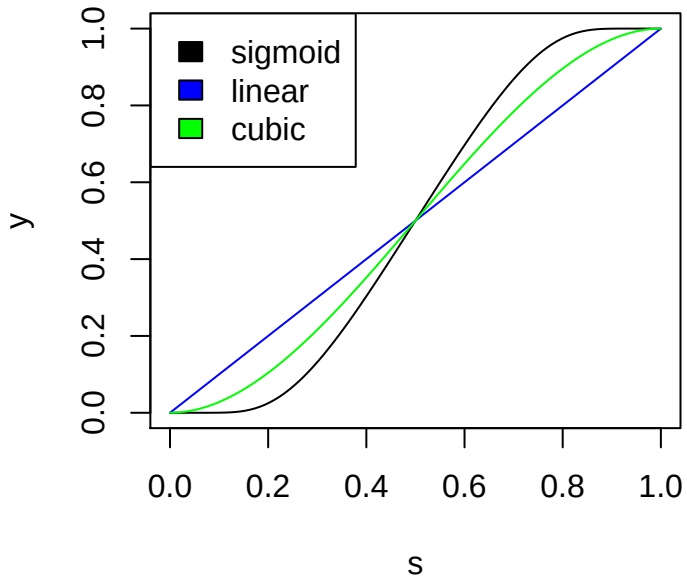


FIGURE 3. Blending functions

When the points are monotonic, i.e. $\operatorname{sgn}(v_-) \neq \operatorname{sgn}(v_+)$ the coefficients a, b and c are computed as

$$\begin{aligned} a &= \frac{2v_-v_+}{v_- + v_+} \\ b &= 0 \\ c &= \frac{v_- + v_+}{v_- - v_+} \end{aligned} \tag{8}$$

For non-monotonic triples, we have

$$\begin{aligned} a &= -2v_-v_+\frac{v_- + v_+}{(v_- - v_+)^2} \\ b &= 2\frac{v_-v_+}{v_- - v_+} \\ c &= \frac{v_- - v_+}{v_- + v_+} \end{aligned} \tag{9}$$

3.1. **Blending and degree.** There is a basis function in every knot, so we glue the functions f_1 and f_2 together as a convex combination $tf_1(x) + (1-t)f_2(x-1)$ ($x-1$ is the normalized coordinate for f_2, the basis function in the knot x_+). We use a linear blender, $t = 1 - x$. Alternatively we could modify the t with a sigmoid blender like $t \in [0, 1/2] \mapsto \exp(2 - 1/t)/2$ and $t \in (1/2, 1] \mapsto 1 - \exp(2 - 1/(1-t))/2$. A cubic blender is also available. These can be selected by the argument `blend="linear"`, `blend="sigmoid"`, or `blend="cubic"` to the stalker interpolant. The three blending functions are shown in figure 3.

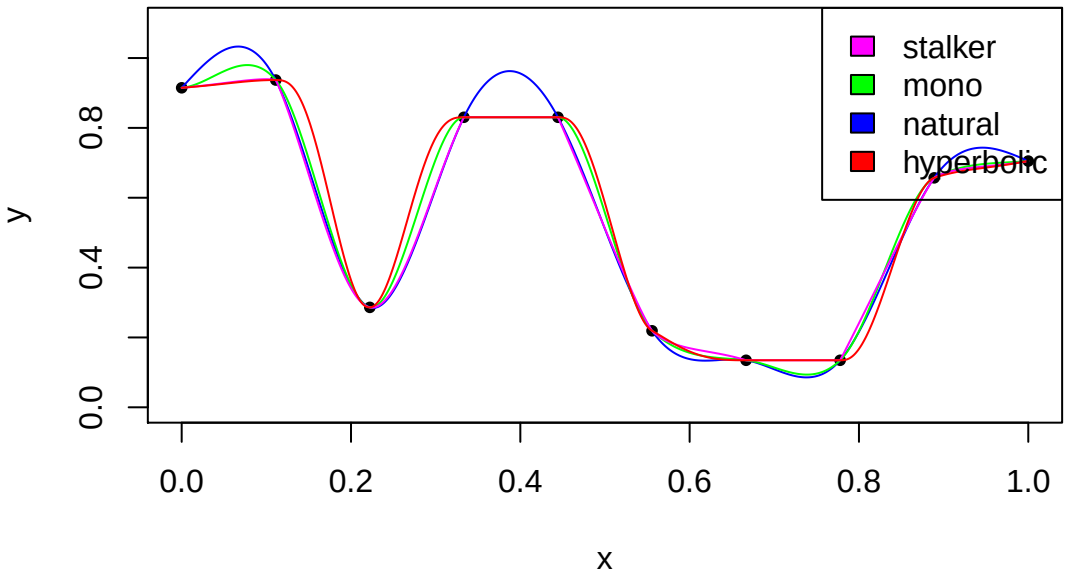
```
sigmoid <- function(t) ifelse(t<0.5, 0.5*exp(2-1/t), 1-0.5*exp(2-1/(1-t)))
cubic <- function(t) -2*t^3 + 3*t^2
linear <- function(t) t
s <- seq(0,1,length=100)
plot(s,sigmoid(s),typ='l',ylab='y')
lines(s,linear(s), col='blue')
lines(s,cubic(s), col='green')
legend('topleft',legend=c('sigmoid','linear','cubic'),fill=c('black','blue','green'))
```

The interpolant can be given a `"degree"` argument, this can be a vector, a fixed degree for each dimension.

In more than one dimension, torsion can be a problem for stability. It can create striping artefacts. This can be alleviated by having constant degree (different from `NA`), except possibly for the first dimension. We'll see an example later.

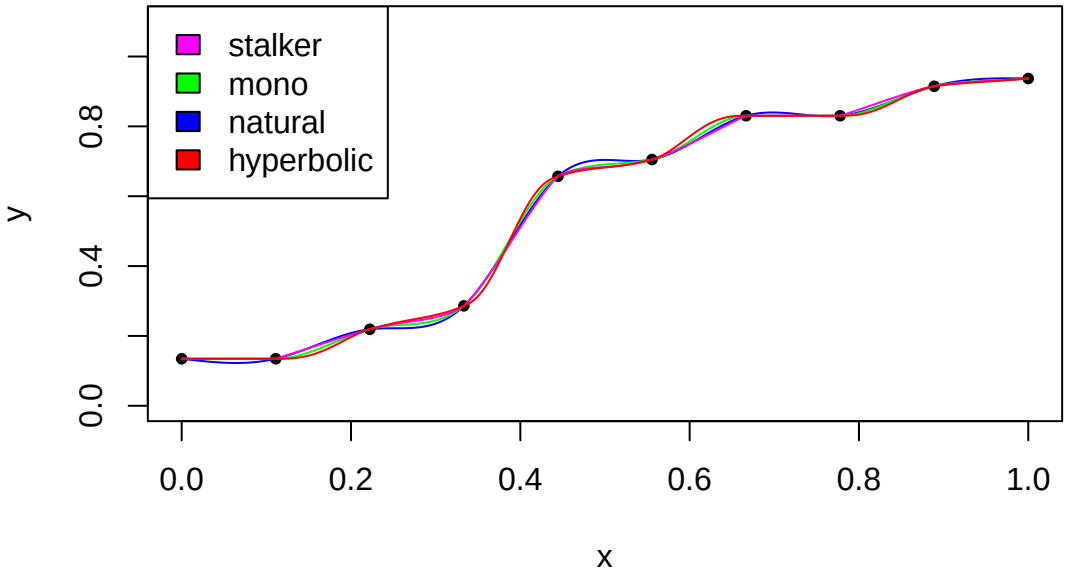
4. Examples

In this section we compare the stalker spline to the `"natural"` and `"monoH.FC"` spline from `stats::splinefun`. We also illustrate the hyperbolic stalker spline with a cubic blender. With the hyperbolic stalker, the linear blender is not able to smooth out the pole in the flat case, whereas the cubic is. The sigmoid blender will smooth out any non-essential singularity, but the resulting curve may not be very pleasant. Plotting the derivative of these interpolants is not for the faint-hearted.

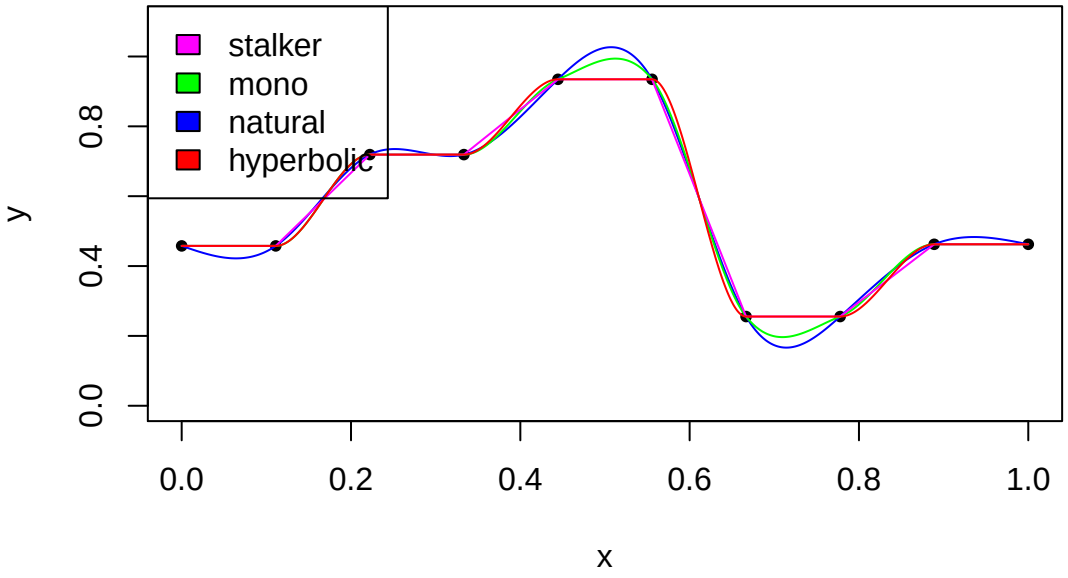


Note that both the stalker and the `"monoH.FC"` spline honours the completely flat region between points 4 and 5, but between points 7 and 8 `"monoH.FC"` has considerable overshoot, even though the points are very close. The reason is that point 8 is slightly lower than point 7, so that points 7-10 are not monotonic, and then the spline there abandons its monotonicity constraint entirely. Mathematically, the stalker spline is differentiable except in points 4 and 5, even though it looks like a sharp corner in point 8 due to a very large second derivative.

We also illustrate the same splines on a monotonic set of points. If all the knots are monotonic, the Fritsch-Carlson spline is superb, it ensures monotonicity and differentiability. The stalker spline does not in case there are completely flat regions, then differentiability is abandoned.



An interesting case is when the knots are pairwise constant. The stalker spline reduces to a linear interpolation. The knots below are not exactly pairwise constant, they differ by 10^{-16}. This is sufficient to make `"monoH.FC"` overshoot.



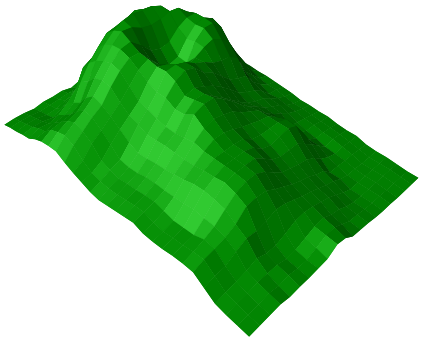
5. Higher dimensions

The stalker spline in higher dimension N is simplistic and problematic. It works on a Cartesian grid. When evaluating the stalker in an x between grid points, the stalker is evaluated for the $N-1$ first dimensions on two grid lines on each side of x in dimension N. On these four points in dimension N, two basis functions are found, evaluated and blended. There is no guarantee that doing this with the dimensions in another order will yield exactly the same result.

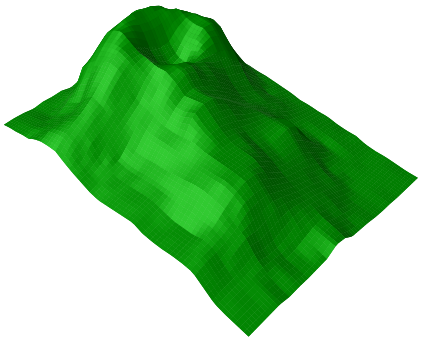
Everything comes at a price, the price for the stalker spline is that it does not work very well in higher dimensions. The variable degree causes creasing artefacts.

We take a look at 2d-interpolation, first the Maungawhau volcano with exaggerated height in figure 4. It is quite nice with the stalker interpolation.

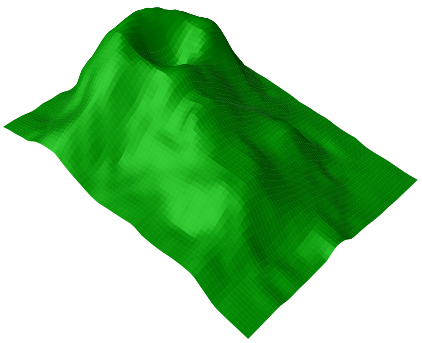
```
data(volcano)
volc <- volcano[seq(1,nrow(volcano),3),seq(1,ncol(volcano),3)]/10 #low res volcano
grid <- list(x=as.numeric(seq_len(nrow(volc))), y=as.numeric(seq_len(ncol(volc))))
ph <- ipol(volc, grid=grid, method='polyharmonic',k=2)
st <- ipol(volc, grid=grid, method='stalker',k=NA)
ml <- ipol(volc, grid=grid, method='multilinear')
g <- list(x=seq(1,nrow(volc), len=71), y=seq(1,ncol(volc),len=71))
par(mar=rep(0,4)); col <- 'green'
light <- list(specular=0.2,ambient=0.0,diffuse=0.6)
plot3D::persp3D(grid$x, grid$y, volc, colvar=NULL, lighting=light,
        theta=45, ltheta=0, lphi=40, col=col, axes=FALSE, bty='n',scale=FALSE)
for(f in list(ml, st, ph)) {
  plot3D::persp3D(g$x, g$y, evalongridV(f,grid=g), colvar=NULL, lighting=light,
        theta=45, ltheta=0, lphi=40, col=col, axes=FALSE, bty='n', scale=FALSE)
}
```



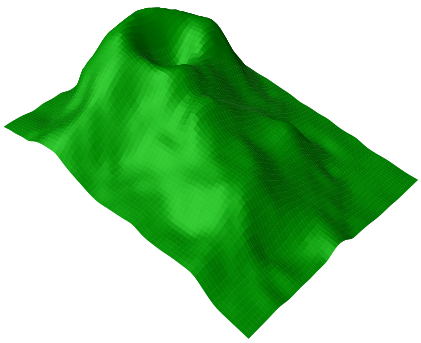
(A) low resolution



(B) multilinear



(C) stalker



(D) thin plate spline

FIGURE 4. Maungawhau

Then we interpolate some random points in figure 5. Incidentally, there are some plane areas, and some torsion. Torsion is a problem for the stalker spline, the degree in one dimension goes down to 1, creating striped artefacts in some regions. This is alleviated by using a constant degree. The hyperbolic stalker is remarkably faithful to the points, but it also suffers from artefacts in certain areas where it is ill-conditioned. There is no advanced shading in `plot3D::persp3D`, so the 100×100 resolution can be seen if you zoom in.

```
set.seed(42); N <- 8
grid <- list(x=seq(0,1,length=N)+c(0,rnorm(N-2,sd=0.3/N),0),
             y=seq(0,1,length=N)+c(0,rnorm(N-2,sd=0.3/N),0))
val <- matrix(runif(N*N,0,0.3),N)
st <- ipol(val,grid=grid, method='stalker',k=NA)
ph <- ipol(val,grid=grid, method='polyharmonic', k=2)
fh <- ipol(val,grid=grid, method='fh', k=0)
sthyp <- function(x) st(x,deg=0,blend='cubic')
g <- list(x=seq(0,1, len=70), y=seq(0,1,len=70))
par(mar=rep(0,4))
for(f in list(st, ph, sthyp, fh)) {
  plot3D::persp3D(g$x, g$y, evalongridV(f,grid=g), colvar=NULL, lighting=light,
         theta=60, ltheta=30, lphi=45, col='green', axes=FALSE, bty='n', scale=FALSE,zlim=c(0,1))
  pts <- evalongridV(f,grid=grid)+0.00
  plot3D::points3D(rep(grid$x,N),rep(grid$y,each=N),pts,add=TRUE,colvar=NULL,pch=20)
}
```

The torsion problem can be seen near the right corner. We can take a closer look at these portions of the surface in figure 6. The striped artefacts are regions where the degree changes.

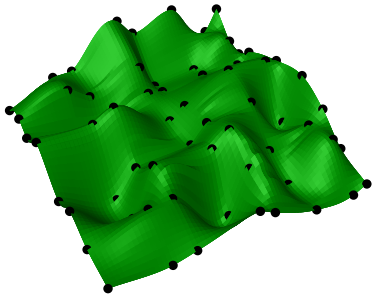
```
g <- list(x=seq(grid$x[5],grid$x[8],length=100),y=seq(grid$y[5],grid$y[8],length=100))
par(mar=rep(1,4))
plot3D::persp3D(g$x, g$y, evalongridV(st,grid=g), colvar=NULL, lighting=light,
       theta=120, ltheta=100, lphi=45, col='green', axes=TRUE)
subgrid <- list(x=grid$x[5:8],y=grid$y[5:8])
zval <- evalongridV(f,grid=subgrid)+0.005
plot3D::points3D(rep(subgrid$x,4),rep(subgrid$y,each=4),zval,add=TRUE,colvar=NULL,pch=20)
```

What happens here is the following. The spline along the rightmost border ($y = 1$) is monotonic. The spline next to it has a top, and at some point it crosses the level of the rightmost level. As we remember, a spline along the other dimension will then reduce its degree all the way down to piecewise linear. It is in this process the lowering of the degree on the right side causes reversal of the derivative between the grid points. We can see it clearly in figure 7 if we plot the splines for the four y values, and the black one for a spline in between. The grey curve is the one with constant degree 2 in the y dimension.

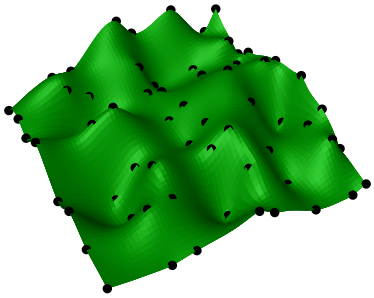
```
x <- seq(grid$x[[5]],grid$x[[8]],len=100)
pts <- c(grid$y[5:8])
pt <- mean(grid$y[6:7])
col <- c('green','magenta','blue','red')
plot(x,st(rbind(x,pt)),typ='l',ylim=c(0,0.3),ylab='z')
lines(x,st(rbind(x,pt),deg=c(1.5,2)), col='grey')
points(grid$x,st(rbind(grid$x,pt)),pch=20)
for(i in seq_along(pts)) {
  v <- st(rbind(x,pts[i]))
  lines(x,v,col=col[i],lty=2)
  points(grid$x,st(rbind(grid$x,pts[i])),pch=20,col=col[i])
}
legend('topright',title='y=',
    legend=round(c(pts[1:2],pt,pts[3:4]),3),
    fill=c(col[1:2],'black',col[3:4]))
```

Note that the black curve is always between the blue and the magenta, but erratic wherever any of the coloured lines cross.

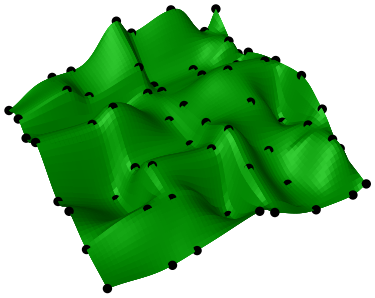
In figure 8 is the same part of the surface with hyperbolic stalker in the first dimension, and constant degree 1.5 in the second.



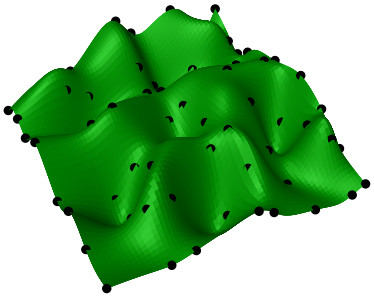
(A) stalker



(B) thin plate spline



(C) hyperbolic stalker



(D) Floater-Hormann

FIGURE 5. Random surface

```
par(mar=rep(1,4))
plot3D::persp3D(g$x, g$y, evalongridV(st,grid=g,degree=c(0,1.5)), colvar=NULL, lighting=light,
       theta=120, ltheta=100, lphi=45, col='green', axes=TRUE)
plot3D::points3D(rep(subgrid$x,4),rep(subgrid$y,each=4),zval,add=TRUE,colvar=NULL,pch=20)
```

6. SUMMARY

The stalker spline is created and used with

```
st <- ipol(val,grid=grid,method='stalker',k=1.5)
st(x,degree=1.2,blend='linear')
```

where k is the degree. `k` can be a vector, one degree for each dimension. It is possible to specify any nonzero degree, though I see no use for degrees less than 1 other than for artful displays. A degree of zero is treated specially, with hyperbolic functions $a+bx+\frac{d}{1+cx}$. Specifying `k=NA` makes the spline adjust the degree locally as described above. The hyperbolic and varying degree stalker do not generalize very well to two or more dimensions, so for multidimenional data varying degree or hyperbolic stalker is strongly discouraged, except possibly in the first dimension. Evaluation

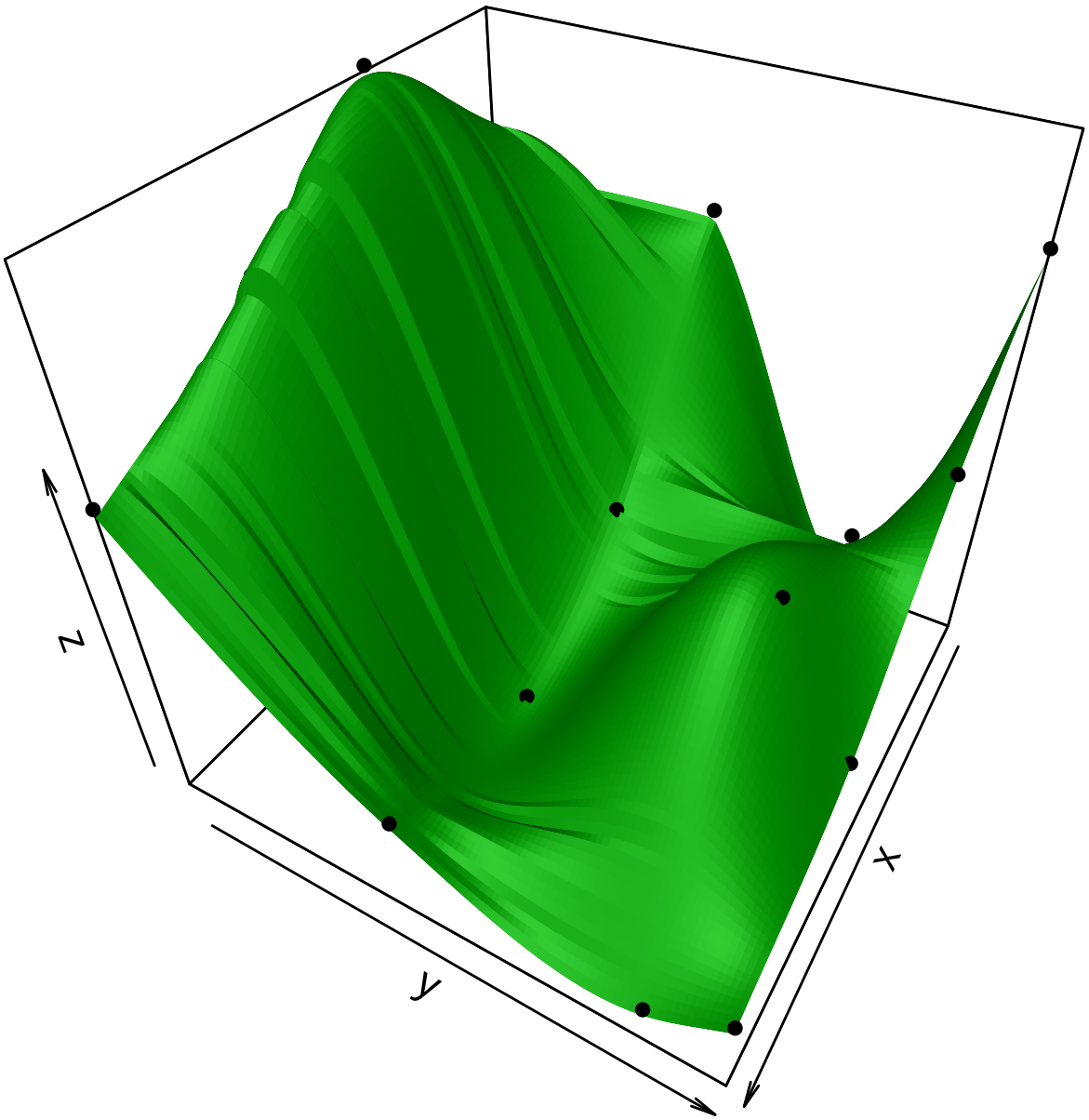


FIGURE 6. Torsion problem in the wet paper spline

on a non-uniform grid currently involves solving a non-linear equation numerically, so this is slower than on a uniform grid.

The spline is created with a default degree, be it `NA` or a numeric between 1 and 2. It is possible at evaluation time to use a different degree, this incurs a time penalty. Ordinarily, basis functions are combined linearly. I.e. when we approach a grid point more and more of the basis function living there is weighted in. This can be changed at evaluation time, it can be done "faster" with a sigmoid map, i.e. so that near a grid point, the neighbouring basis functions are not used at all. Use `blend="sigmoid"` or `blend="cubic"` to choose between two such sigmoid maps.

RAGNAR FRISCH CENTRE FOR ECONOMIC RESEARCH, OSLO, NORWAY

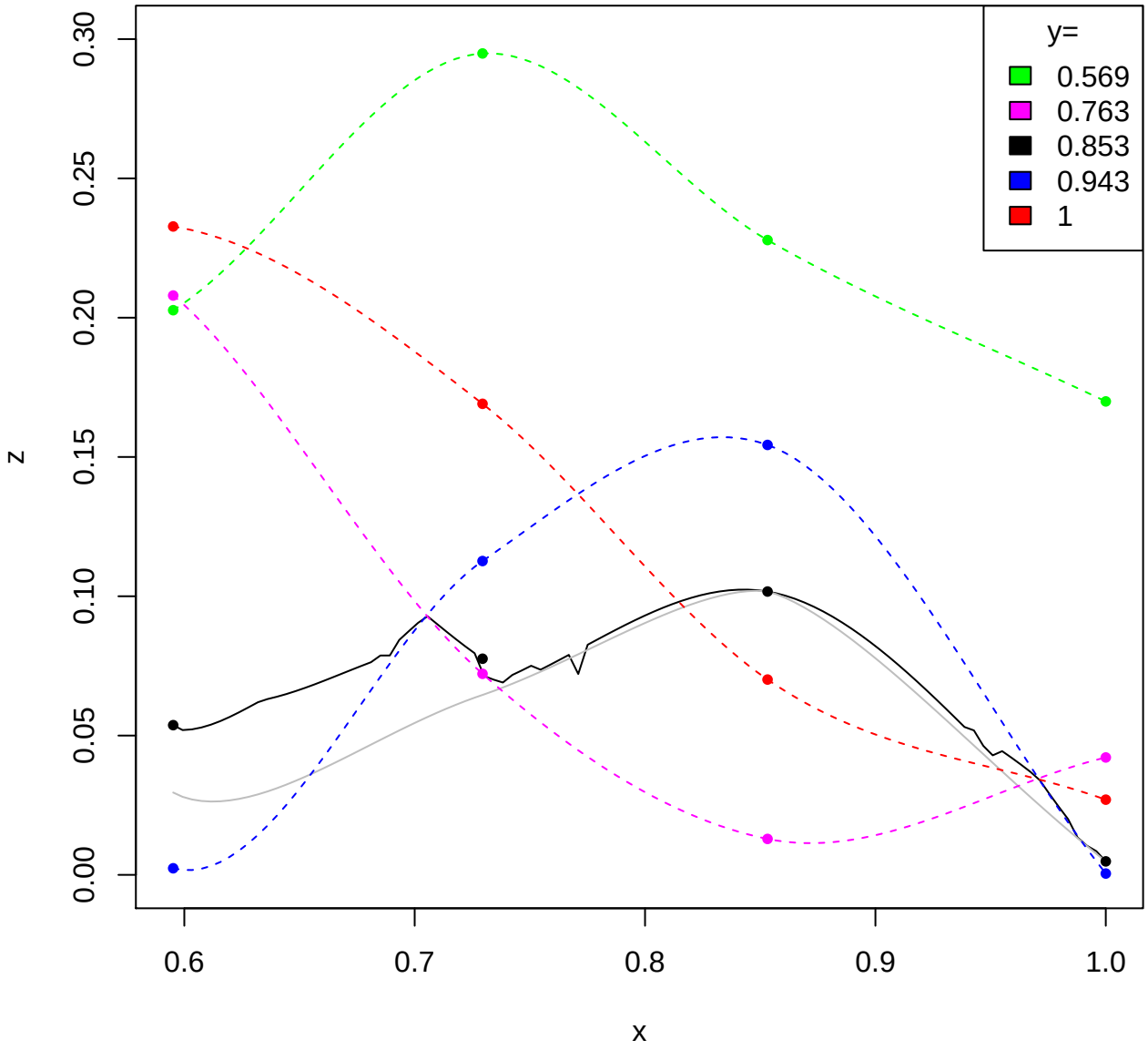


FIGURE 7. Torsion details

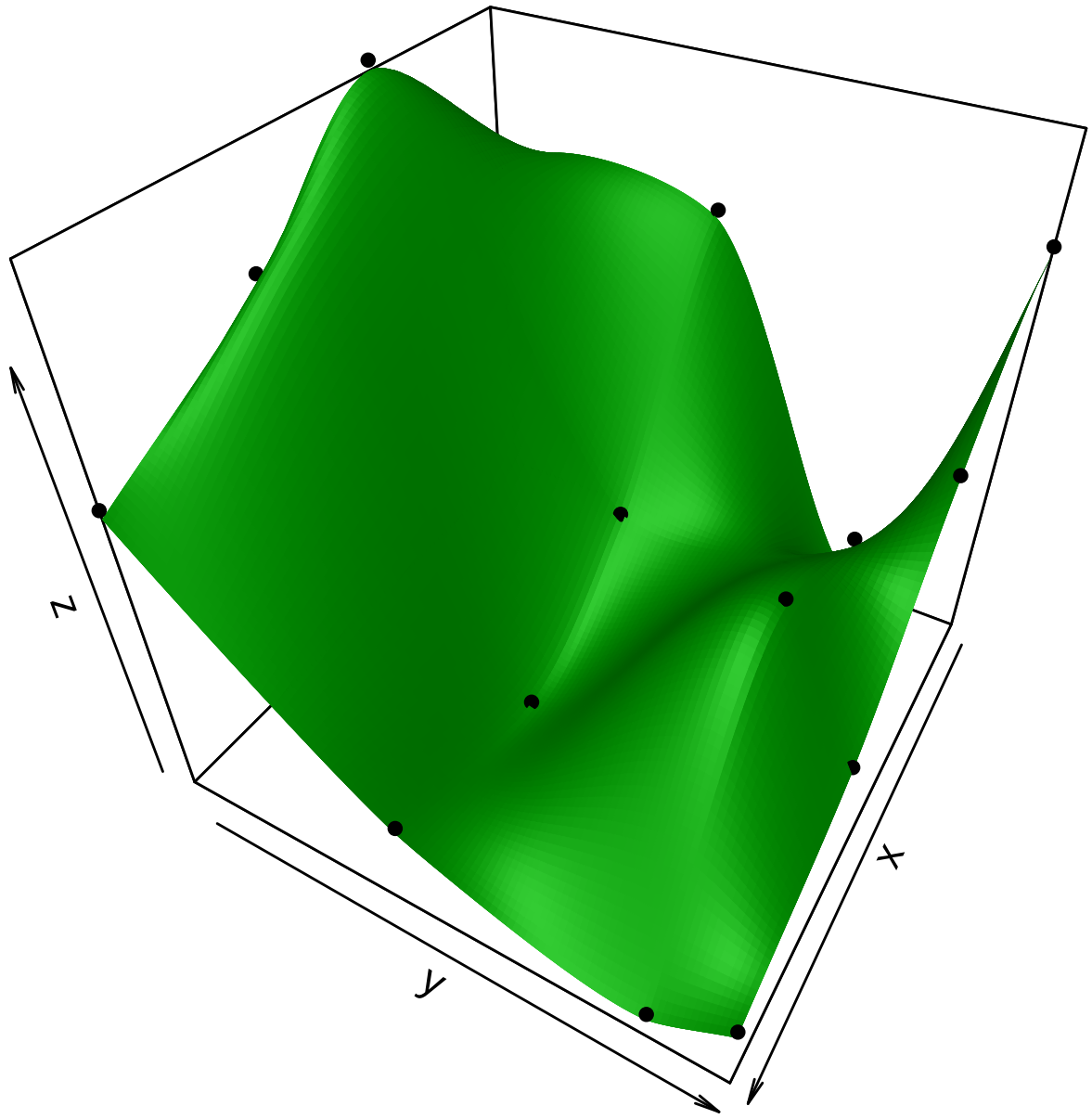


FIGURE 8. Constant degree 1.5