smooth constraint functions

Ben Bolker

Often when we're fitting models, we have functions that go somewhere they're not supposed to (e.g. densities that fall below 0 or probabilities that exceed the range 0<x<1). If we can parameterize everything so that the constraints apply to a single parameter, we can use *box constraints* or we can reparameterize the model in terms of an unconstrained parameter (e.g. fit on the scale of $\log(\beta)$ for a value that needs to be positive, or on the scale of $\text{logit}(\beta)$ for a value that needs to be between 0 and 1. Sometimes, however, it's not easy to see how to reparameterize the model/constrain a single parameter. A useful strategy in this case is to force the value *smoothly* to its boundary, and at the same time add a quadratic penalty that penalizes the objective function for failing to stay inside the bounds.

AD Model Builder (Fournier et al. 2011), a powerful tool especially used in fisheries modeling, defined `posfun` as follows:

$$f(x) = \begin{cases} \frac{\epsilon}{2-x/\epsilon}, & \text{if } x < \epsilon \\ x & \text{otherwise} \end{cases}$$

(`posfun` also had the side effect of adding $\gamma(x-\epsilon)^2$ to a running penalty term if $x < \epsilon$; the accumulated penalty could be added to the negative log-likelihood.)

This function has been widely used in applied fisheries research (Breen et al. 2003; Branch and Hilborn 2010; Carruthers, McAllister, and Taylor 2011; Rudd and Branch 2017). It has the following useful properties: for a given $\epsilon > 0$,

- $f(x) = x$ for $x > \epsilon$
- $f(x) > 0$ for all x
- $f'(x) > 0$ for all x
- $f(x) = \epsilon/2$ for $x = 0$

However, it only has a continuous *first* derivative at $x = \epsilon$. This causes problems if we are trying to do any numerical operations that depend on a continuous second derivative, e.g. Laplace approximation (or Riemannian Hamiltonian Monte Carlo … (Girolami, Calderhead, and Chin 2019)).

Thus we need a function $f(x)$ that also satisfies

- $f(x)$, $f'(x)$, and $f''(x)$ are everywhere continuous ;

this implies $f(\epsilon = \epsilon)$; $f'(\epsilon) = 1$; $f''(\epsilon) = 0$. (We are willing to give up the last property above ($f(0) = \epsilon/2$).

Start by setting $x' = (\epsilon - x)$. Then $x' > 0$ for $x < \epsilon$ and $x' = 0$ when $x = \epsilon$. Suppose we take $g(x') = \left(1 + ax' + bx'^2\right)$ and $f(x') = \epsilon g(x')^{-1}$. $g(x' = 0) = 1$, so $f(x' = 0) = \epsilon$. Now

$$f'(x' = 0) = -\epsilon g'(0)(g(0))^{-2} = -\epsilon a$$

$$
\begin{aligned}
f''(x'=0) &= -\epsilon\left(g''(0)(g(0))^{-2} + g'(0)\cdot -2g'(0)(g(0))^{-3}\right) \\
&= -\epsilon\left(\frac{g''(0)-2(g'(0))^2 g(0)}{g(0)^2}\right) \\
&= -\epsilon\left(2b-2a^2\right)
\end{aligned}
$$

So we need $a=-1/\epsilon$, $b=1/\epsilon^2$?

Let's test it:

```
f <- function(x,eps=0.001) {
    eps*(1/(1-(x-eps)/eps + (x-eps)^2/eps^2))
}
f(0.001)
```

```
## [1] 0.001
```

```
library(numDeriv)
grad(f,0.001)
```

```
## [1] 1
```

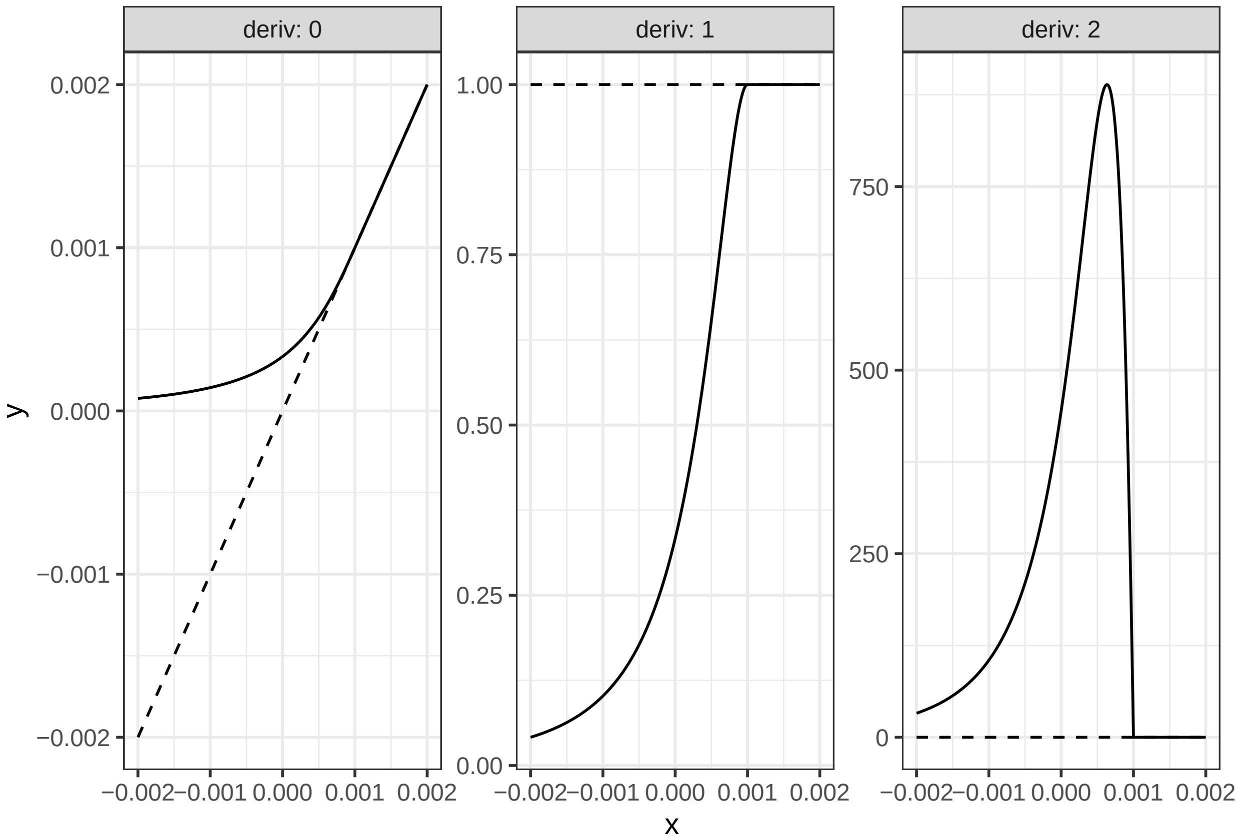
```
## not exactly zero but close enough ...
all.equal(drop(hessian(f,0.001)),0)
```

```
## [1] TRUE
```

Can we figure out what the general form would be to make all higher derivatives zero? Does this Taylor series converge to something easily recognizable ... ?

```
xvec <- seq(-0.002,0.002,length=601)
dfun <- function(f,xvec,eps=0.001,n) {
    gval <- switch(as.character(n), "0"=xvec, "1"=1, 0)
    e <- body(f)[[2]]
    for (i in seq_len(n)) {
        e <- D(e,name="x")
    }
    lval <- eval(e,list(x=xvec))
    return(ifelse(xvec<eps,lval,gval))
}

mkderivs <- function(f,maxd=3) {
    return(purrr::map_dfr(setNames(0:maxd,0:maxd),
                          ~tibble::tibble(x=xvec,y=dfun(f,xvec,n=.)),
                          .id="deriv"))
}

mkcomp <- function(maxd=3) {
    return(purrr::map_dfr(setNames(0:maxd,0:maxd),
                          ~tibble::tibble(x=xvec,
                                          y=dplyr::case_when(.==0 ~ xvec,
                                                             .==1 ~ 1,
                                                             TRUE ~ 0)),
                          .id="deriv"))
}
```

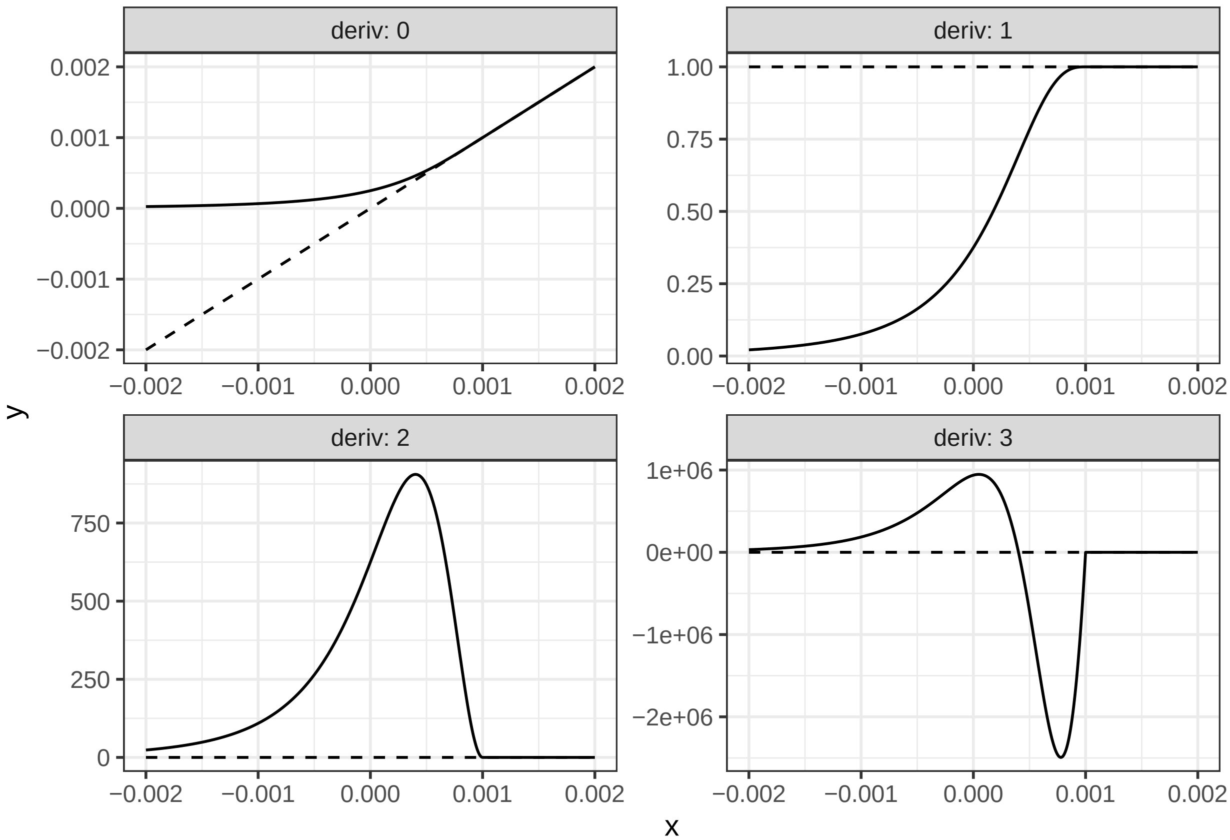
```r
library(ggplot2); theme_set(theme_bw())

plotfun <- function(f,maxd=2,xvec=xvec) {
    dd <- mkderivs(f,maxd)
    cc <- mkcomp(maxd)
    return(ggplot(dd, aes(x,y))
           + geom_line()
           + geom_line(data=cc,linetype=2)
           + facet_wrap(~deriv,scale="free",
                        labeller=label_both)
           )
}

print(plotfun(f))
```



What if we went one more step (i.e. make $g(x') = \left(1 + ax' + bx'^2 + cx^3\right)$?)

Tried to do the algebra myself but Wolfram Alpha does it better: `Solve[D[D[D[eps/(1-x/eps+x^2/eps^2 + c x^3),x],x],x]==0, {c}]`

```r
f <- function(x,eps=0.001) {
    eps*(1/(1-(x-eps)/eps + (x-eps)^2/eps^2 - (x-eps)^3/eps^3))
}
print(plotfun(f,3))
```

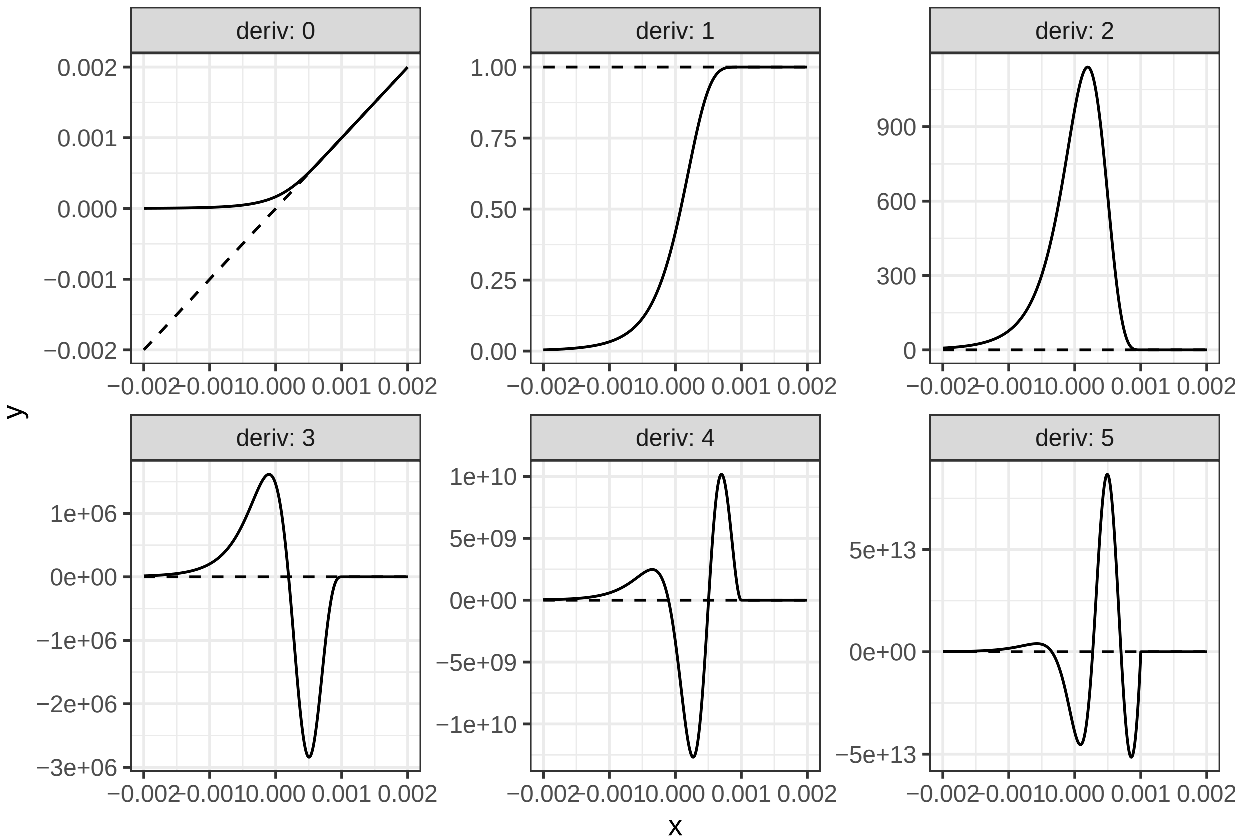


By induction/guessing, we have

$$f(x') = \epsilon \left(\sum_{i=0}^{n} (-1)^i \left(x'/\epsilon\right)^i \right)$$

(what is this function?)

```
f <- function(x,eps=0.001) {
    eps*(1/(1-(x-eps)/eps + (x-eps)^2/eps^2 - (x-eps)^3/eps^3
        + (x-eps)^4/eps^4 - (x-eps)^5/eps^5))
}
print(plotfun(f,5))
```



Note that while the derivatives are indeed continuous, the magnitudes increase dramatically as we go to higher orders; this could conceivably cause problems for very sensitive problems … ?

References

Branch, Trevor A., and Ray Hilborn. 2010. "A General Model for Reconstructing Salmon Runs." *Canadian Journal of Fisheries and Aquatic Sciences* 67 (5): 886–904.

Breen, Paul A, Ray Hilborn, Mark N Maunder, and Susan W Kim. 2003. "Effects of Alternative Control Rules on the Conflict Between a Fishery and a Threatened Sea Lion (*Phocarctos Hookeri*)." *Canadian Journal of Fisheries and Aquatic Sciences* 60 (5): 527–41. doi:10.1139/f03-046.

Carruthers, Thomas R., Murdoch K. McAllister, and Nathan G. Taylor. 2011. "Spatial Surplus Production Modeling of Atlantic Tunas and Billfish." *Ecological Applications* 21 (7): 2734–55.

Fournier, David A., Hans J. Skaug, Johnoel Ancheta, James Ianelli, Arni Magnusson, Mark N. Maunder, Anders Nielsen, and John Sibert. 2011. "AD Model Builder: Using Automatic Differentiation for Statistical Inference of Highly Parameterized Complex Nonlinear Models." *Optimization Methods and Software*, October, 1–17. doi:10.1080/10556788.2011.597854.

Girolami, Mark, Ben Calderhead, and Siu A. Chin. 2019. "Riemannian Manifold Hamiltonian Monte Carlo." *arXiv:0907.1100 [Cs, Math, Stat]*, December. http://arxiv.org/abs/0907.1100.

Rudd, Merrill B., and Trevor A. Branch. 2017. "Does Unreported Catch Lead to Overfishing?" *Fish and Fisheries* 18 (2): 313–23.