Examples of usage

Pablo Rodríguez-Sánchez

2019-03-04

Installation

To install the package, please type

`devtools::install_github("PabRod/consRvative", ref = "develop")`

in your *R* console.

One dimensional examples

Allee effect

A single-species population dynamics model with Allee effect is governed by the following differential equation:

$$\frac{dN}{dt} = rN\left(\frac{N}{A} - 1\right)\left(1 - \frac{N}{K}\right)$$

It is easy to see that this differential equation has three equilibrium points, $N = 0$, $N = K$ and $N = A$, being all of them stable but the latter one, which is unstable. We'll use the parameters $r = 1$, $A = 0.5$ and $K = 1$.

```r
r <- 1
A <- 0.5
K <- 1

f <- function(x) { r * x * (x/A - 1) * (1 - x/K) }
```

We can use our method `approxPot1D` to approximate the potential function at a set of points. First, we have to create the points.

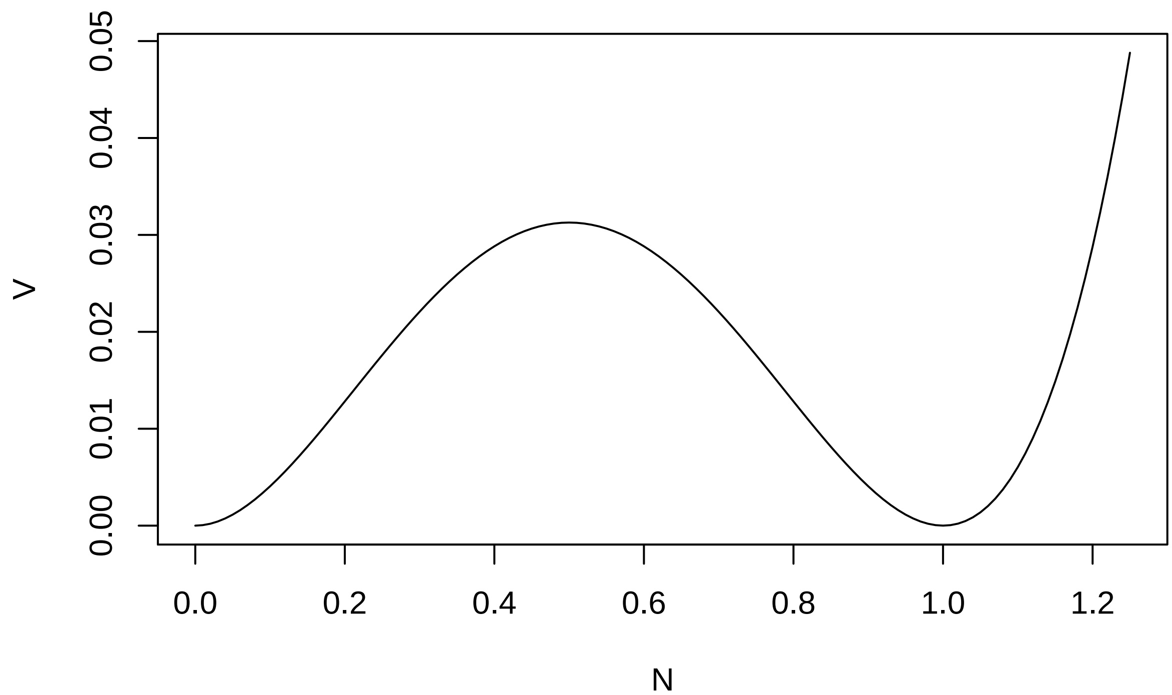
```r
xs <- seq(0, 1.25, by = 0.01)
```

and then pass them to our algorithm:

```r
Vs <- approxPot1D(f, xs)
```

By plotting the result, we clearly see that the two stable equilibria appear at $N = 0$ and $N = K = 1$, and the unstable one at $N = A = 0.5$, as we expected.

```r
plot(xs, Vs,
     type = 'l', xlab = 'N', ylab = 'V')
```



Two dimensional examples

Four well potential

In this section we'll deal with the two-dimensional differential equation given by:

$$\begin{cases} \frac{dx}{dt} = f(x,y) = -x(x^2-1) \\ \frac{dy}{dt} = g(x,y) = -y(y^2-1) \end{cases}$$

This is a gradient system, that can be derived from the potential:

$$V(x,y) = \frac{x^2}{4}(x^2-2) + \frac{y^2}{4}(y^2-2) + V_0$$

It is important to remember that we'll code the function's input as a vector:

```r
f <- function(x) {c(-x[1]*(x[1]^2 - 1),
                    -x[2]*(x[2]^2 - 1))}
```

Our region of interest is now two-dimensional. We need, thus, two vectors to create our grid of points:

```
xs <- seq(-1.5, 1.5, by = 0.05)
ys <- seq(-1.5, 1.5, by = 0.05)
```

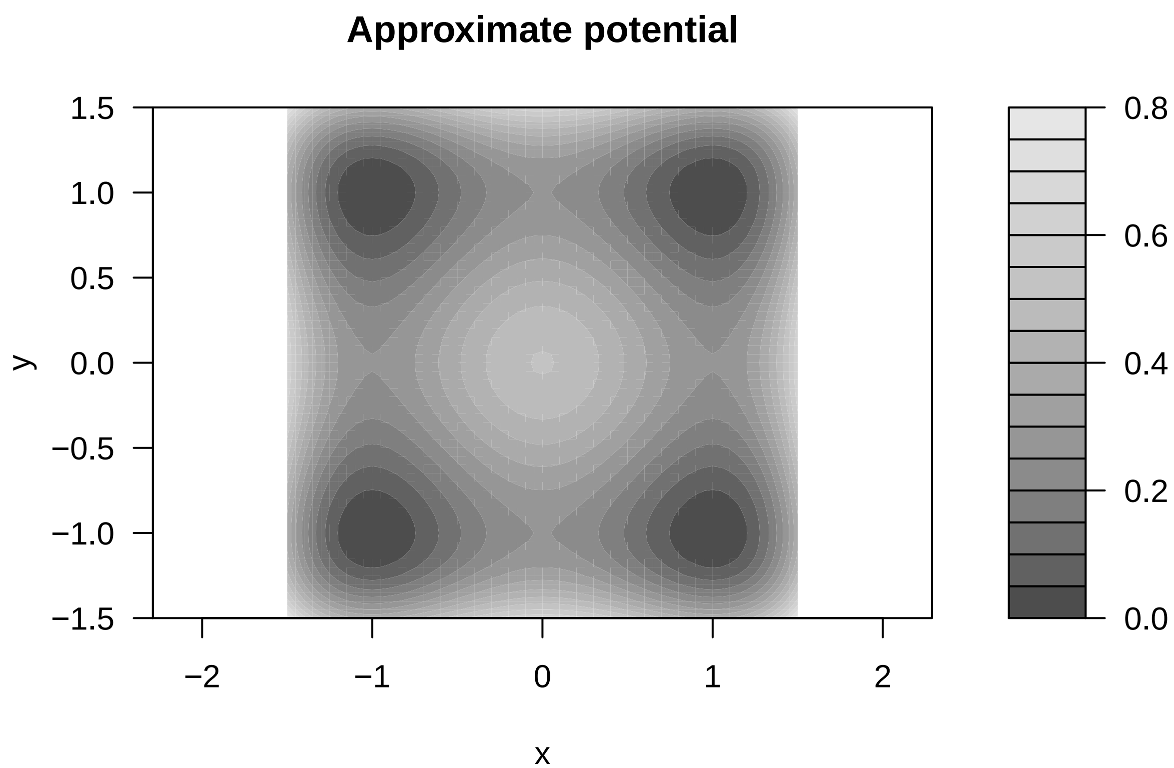
Now we are ready to apply `approxPot2D`:

```
result <- approxPot2D(f, xs, ys)
```

`result` is a list that contains two fields:

- `result$V` contains the estimated values of the potentials at each grid point
- `result$err` contains the estimated error at each grid point

```
filled.contour(xs, ys, result$V,
        color=gray.colors, asp = 1,
        main = 'Approximate potential', xlab = 'x', ylab = 'y')
```



Provided our example is a gradient system, we expect our approximation error to be zero everywhere.

```
max(result$err)
```

```
## [1] 0
```

Simple regulatory gene network

A bistable network model can be described by a set of equations of the form:

$$\begin{cases} \frac{dx}{dt} = f(x,y) = b_x - r_x x + \frac{a_x}{k_x + y^n} \\ \frac{dy}{dt} = g(x,y) = b_y - r_y y + \frac{a_y}{k_y + x^n} \end{cases}$$

We can code it in vector form:

```
# Parameters
bx <- 0.2
ax <- 0.125
kx <- 0.0625
rx <- 1

by <- 0.05
ay <- 0.1094
ky <- 0.0625
ry <- 1

n <- 4

# Dynamics
f <- function(x) {c(bx - rx*x[1] + ax/(kx + x[2]^n),
                    by - ry*x[2] + ay/(ky + x[1]^n))}
```

This set of equations is, in general, not gradient (because $\frac{\partial f}{\partial y} \neq \frac{\partial g}{\partial x}$). Anyways, we can use the method `approxPot2D` to compute the approximate potential.

First, we need to define our region of interest:

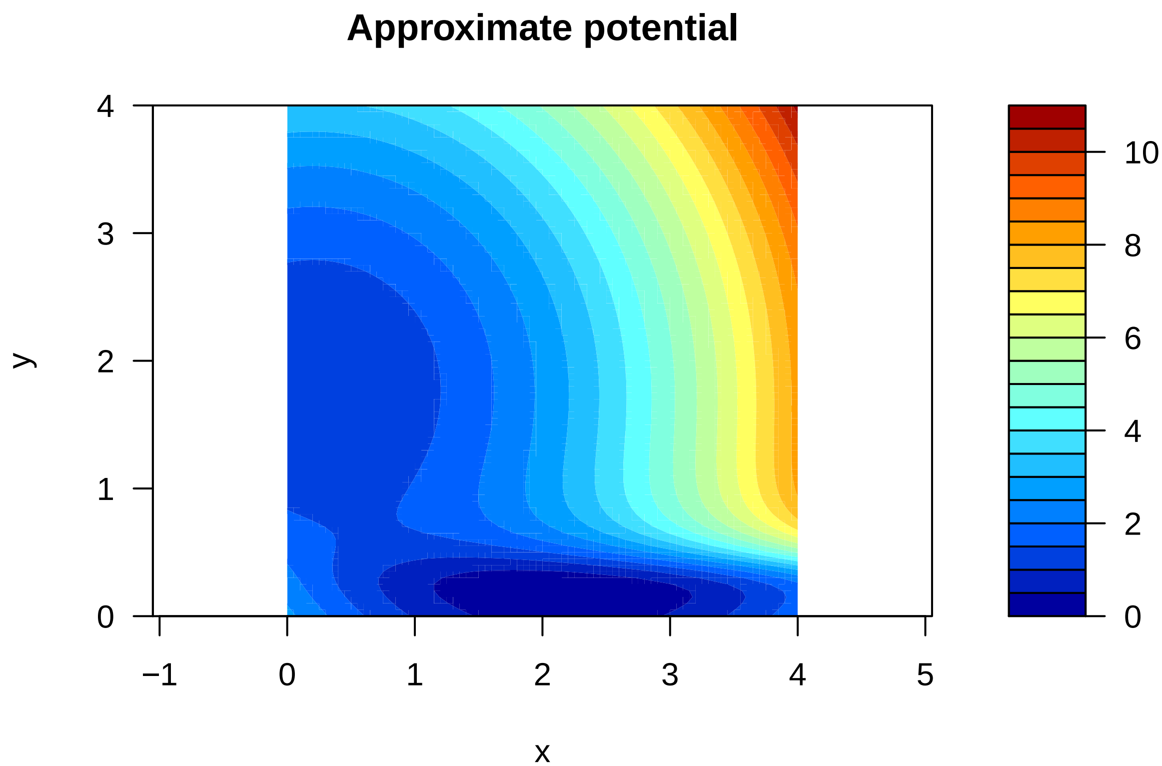
```
xs <- seq(0, 4, by = 0.05)
ys <- seq(0, 4, by = 0.05)
```

And then we are ready to apply our algorithm:

```
result <- approxPot2D(f, xs, ys)
```

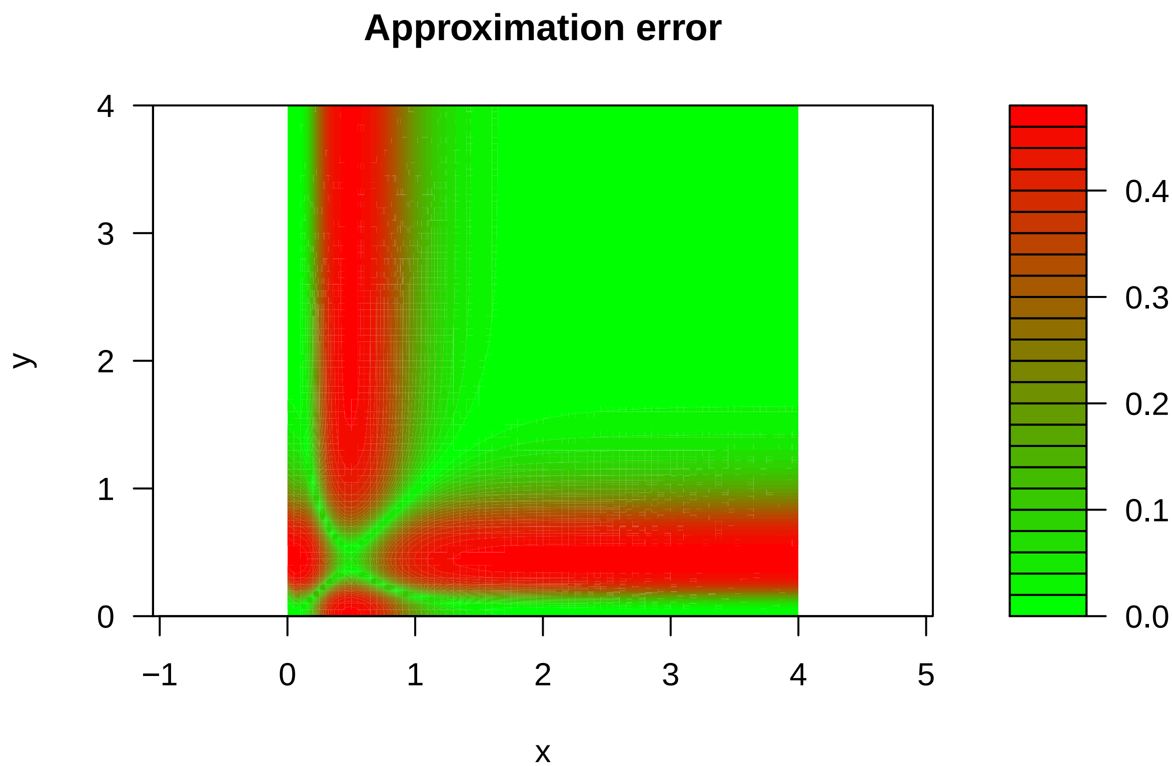
The resulting approximate potential looks like:

```
filled.contour(xs, ys, result$V,
        color = colorRamps::matlab.like, asp = 1,
        main = 'Approximate potential', xlab = 'x', ylab = 'y')
```



Being this not a gradient system it is advisable to plot the estimated error. The areas in green represent small approximation error, so the potential can be safely used in those regions.

```
filled.contour(xs, ys, result$err,
        color = colorRamps::green2red, asp = 1,
        main = 'Approximation error', xlab = 'x', ylab = 'y')
```


Approximation error
y
x
4
3
2
1
0
−1
0
1
2
3
4
5
0.4
0.3
0.2
0.1
0.0