Package 'RootsExtremaInflections'

July 15, 2025

Type Package

Title Finds Roots, Extrema and Inflection Points of a Curve

Version 1.2.5

Date 2025-07-15

Maintainer Demetris T. Christopoulos <dchristop@econ.uoa.gr>

Description Implementation of Taylor Regression Estimator (TRE),
Tulip Extreme Finding Estimator (TEFE), Bell Extreme Finding Estimator (BEFE),
Integration Extreme Finding Estimator (IEFE) and
Integration Root Finding Estimator (IRFE) for roots, extrema and inflections of a curve .
Christopoulos, DT (2019) <doi:10.13140/RG.2.2.17158.32324> .
Christopoulos, DT (2016) <doi:10.2139/ssrn.3043076> .
Christopoulos, DT (2016) <https://demovtu.veltech.edu.in/wp-content/uploads/2016/04/Paper-04-2016.pdf> .
Christopoulos, DT (2014) <doi:10.48550/arXiv.1206.5478> .

License GPL-2

Depends iterators, foreach, parallel, doParallel, inflection

Suggests stats, graphics, grDevices

NeedsCompilation no

Repository CRAN

Date/Publication 2025-07-15 09:00:02 UTC

Author Demetris T. Christopoulos [aut, cre]

Contents

RootsExtremaInflections-package 2
classify_curve 5
extremexi 7
findextreme 9
findmaxbell 11
findmaxtulip 12
findroot 14
inflexi 16

rootexinf 18
rootxi 21
scan_curve 23
scan_noisy_curve 26
symextreme 29
xydat 31

Index **32**

`RootsExtremaInflections-package`
Finds Roots, Extrema and Inflection Points of a Planar Curve

Description

This package contains functions for computing roots, extrema and inflection points of a curve that is the graph of a smooth function when we have only a data set $\{(x_i, y_i), i = 1, 2, \ldots m\}$, generated from it by the procedure $y_i = f(x_i)$ or for the noisy case by $y_i = f(x_i) + \epsilon_i$ with a zero mean error, $\epsilon_i \sim iid(0, \sigma^2)$, by using the *Taylor Regression Estimator (TRE)* method, which is described briefly here. When we want to find a root for a function f by using the traditional Numerical Analysis methods (bisection, secant, Newton-Raphson etc), it is necessary to know the exact formula of f. Unfortunately in research problems we do not know that formula and our data are also of a noisy type. In this package we use the *Taylor Regression Estimator (TRE)* method, which can work when we know the discrete values $\{(x_i, y_i), i = 1, 2, \ldots m\}, y_i = f(x_i)$ of our known or unknown smooth function f. Additionally the method works with satisfactory accuracy also for the corresponding noisy values $\{(x_i, y_i), i = 1, 2, \ldots m\}$, $y_i = f(x_i) + \epsilon_i$, $\epsilon_i \sim iid(0, \sigma^2)$. The computation of extrema and inflection points for a smooth f is merely a problem of root finding for first and second derivative respectively, thus *TRE* method can also find an extreme or an inflection point. In a few words, the method is referencing to the well known Taylor polynomial of a smooth function f around a point ρ,

$$f(x) = a_0 + a_1 (x - \rho) + a_2 (x - \rho)^2 + a_3 (x - \rho)^3 + \ldots + a_n (x - \rho)^n$$

When the coefficients a_0, a_1, a_2, as computed using a polynomial regression, have minimum absolute value, then the corresponding points ρ are the estimations of the *root*, *extreme* or *inflection point*, respectively. Essentially *Taylor Regression (TR)* is *polynomial regression for Taylor polynomial*. For a more rigorous definition of the terms *TR*, *TRE* method, further discussion and numerical examples, see *Christopoulos (2014).*
From version 1.2.1 a set of new methods has been added:
Tulip Extreme Finding Estimator (TEFE)
Bell Extreme Finding Estimator (BEFE)
Integration Extreme Finding Estimator (IEFE)
Integration Root Finding Estimator (IRFE)
All these methods are defined and explained at *Christopoulos (2019).*

Details

After adding functions for finding roots and extrema without regression, current package constitutes the core of new filed , called 'Noisy Numerical Analysis', which combines Geometry, Calculus, Numerical Analysis and Statistics and treat noisy curves for solving known problems of Numerical Analysis.

Author(s)

Demetris T. Christopoulos

Maintainer: Demetris T. Christopoulos <dchristop@econ.uoa.gr>

References

Demetris T. Christopoulos (2014). Roots, extrema and inflection points by using a proper Taylor regression procedure. *SSRN*. doi:10.2139/ssrn.3043076

Demetris T. Christopoulos (2019). New methods for computing extremes and roots of a planar curve: introducing Noisy Numerical Analysis (2019). *ResearchGate*. doi:10.13140/RG.2.2.17158.32324

Examples

```
#Load data:
data(xydat)
#
#Extract x and y variables:
x=xydat$x;y=xydat$y
#
#Find root, plot results, print Taylor coefficients and rho estimation:
b<-rootxi(x,y,1,length(x),5,5,plots=TRUE);b$an;b$froot;
#
#Find extreme, plot results, print Taylor coefficients and rho estimation:
c<-extremexi(x,y,1,length(x),5,5,plots=TRUE);c$an;c$fextr;
#
#Find inflection point, plot results, print Taylor coefficients and rho estimation:
d<-inflexi(x,y,1,length(x),5,5,plots=TRUE);d$an;d$finfl;
# Create a relative big data set...
f=function(x){3*cos(x-5)};xa=0.;xb=9;
set.seed(12345);x=sort(runif(5001,xa,xb));r=0.1;y=f(x)+2*r*(runif(length(x))-0.5);
#
#Find root, plot results, print Taylor coefficients and rho estimation in parallel:
#b1<-rootxi(x,y,1,round(length(x)/2),5,5,plots=TRUE,doparallel = TRUE);b1$an;b1$froot;
# Available workers are 12
# Time difference of 5.838743 secs
#           2.5 %       97.5 %           an
# a0 -0.006960052  0.004414505 -0.001272774
# a1 -2.982715739 -2.933308292 -2.958012016
# a2 -0.308844145 -0.213011162 -0.260927654
# a3  0.806555336  0.874000586  0.840277961
# a4 -0.180720951 -0.161344935 -0.171032943
# a5  0.007140500  0.009083859  0.008112180
# [1] 177.0000000   0.2924279
```

```
# Compare with exact root = 0.2876110196
#Find extreme, plot results, print Taylor coefficients and rho estimation in parallel:
#c1<-extremexi(x,y,1,round(length(x)/2),5,5,plots=TRUE,doparallel = TRUE);c1$an;c1$fextr;
# Available workers are 12
# Time difference of 5.822514 secs
#            2.5 %       97.5 %           an
# a0 -3.0032740050 -2.994123850 -2.998698927
# a1 -0.0006883998  0.012218393  0.005764997
# a2  1.4745326519  1.489836668  1.482184660
# a3 -0.0340626683 -0.025094859 -0.029578763
# a4 -0.1100798736 -0.105430525 -0.107755199
# a5  0.0071405003  0.009083859  0.008112180
# [1] 1022.000000    1.852496
# Compare with exact extreme = 1.858407346
#Find inflection point, plot results, print Taylor coefficients and rho estimation in parallel:
#d1<-inflexi(x,y,1090,2785,5,5,plots=TRUE,doparallel = TRUE);d1$an;d1$finfl;
# Available workers are 12
# Time difference of 4.343851 secs
#           2.5 %       97.5 %            an
# a0 -0.008238016  0.002091071 -0.0030734725
# a1  2.995813560  3.023198534  3.0095060468
# a2 -0.014591465  0.015326175  0.0003673549
# a3 -0.531029710 -0.484131902 -0.5075808056
# a4 -0.008253975  0.007556465 -0.0003487551
# a5  0.016126428  0.034688019  0.0254072236
# [1] 800.000000   3.427705
# Compare with exact inflection = 3.429203673
# Or execute rootexinf() and find a set of them at once and in same time:
#a<-rootexinf(x,y,100,round(length(x)/2),5,plots = TRUE,doparallel = TRUE);
#a$an0;a$an1;a$an2;a$frexinf;
# Available workers are 12
# Time difference of 5.565372 secs
#           2.5 %      97.5 %           an0
# a0 -0.008244278  0.00836885  6.228596e-05
# a1 -2.927764078 -2.84035634 -2.884060e+00
# a2 -0.447136449 -0.30473094 -3.759337e-01
# a3  0.857290490  0.94794071  9.026156e-01
# a4 -0.198104383 -0.17360676 -1.858556e-01
# a5  0.008239609  0.01059792  9.418764e-03
#           2.5 %      97.5 %          an1
# a0 -3.005668018 -2.99623116 -3.000949590
# a1 -0.003173501  0.00991921  0.003372854
# a2  1.482600580  1.50077450  1.491687542
# a3 -0.034503271 -0.02551597 -0.030009618
# a4 -0.115396537 -0.10894117 -0.112168855
# a5  0.008239609  0.01059792  0.009418764
#           2.5 %       97.5 %          an2
# a0  0.083429390  0.092578772  0.088004081
# a1  3.007115452  3.027343849  3.017229650
# a2 -0.009867779  0.006590042 -0.001638868
# a3 -0.517993955 -0.497886933 -0.507940444
# a4 -0.043096158 -0.029788902 -0.036442530
# a5  0.008239609  0.010597918  0.009418764
```

```
#            index     value
# root          74 0.2878164
# extreme      923 1.8524956
# inflection  1803 3.4604842
#Here a first plot always is helpful.
```

`classify_curve` *Classifies a planar curve according to its convexity, symmetry and extreme type*

Description

Given a planar curve we want to know its convexity ranges, the index for use in ESE or EDE methods, the existence of a maximum or minimum and the classification as tulip or bell.

Usage

```
classify_curve(x, y)
```

Arguments

x	A numeric vector for the independent variable without missing values
y	A numeric vector for the dependent variable without missing values

Details

A first check for the existence of infinities is performed. If infinities exist, then all outputs are set to 'NA', otherwise main code works.

Value

A list with next members:

1. ctype the convexity type of curve: convex, concave, convex/concave or concave/convex
2. index the relevant index for ESE or EDE usage: 0,1,0,1 respectively to 'ctype' previously presented values
3. asymmetry a classification of asymmetry type, if exists
4. totalconvexity the overall dominant convexity type of curve if we omit the fact probably has ranges of different types of convexity
5. ismax logical value, TRUE if a maximum seems to exits, FALSE if a minimum seems to be the case, 'NA' otherwise
6. shapetype the shape of extreme as tulip, bell or 'NA'

Note

Results of current function have an approximation type, since not all kind of curves can be classified by a given procedure. Caution has been taken in order to be able to infer for the very basic attributes.

Author(s)

Demetris T. Christopoulos

References

[1]Demetris T. Christopoulos (2019). New methods for computing extremes and roots of a planar curve: introducing Noisy Numerical Analysis (2019). *ResearchGate*. doi:10.13140/RG.2.2.17158.32324

[2]Demetris T. Christopoulos (2014). Developing methods for identifying the inflection point of a convex/concave curve. arXiv:1206.5478v2 [math.NA]. doi:10.48550/arXiv.1206.5478

[3]Demetris T. Christopoulos (2016). On the efficient identification of an inflection point.International Journal of Mathematics and Scientific Computing, (ISSN: 2231-5330), vol. 6(1). https://demovtu.veltech.edu.in/wp-content/uploads/2016/04/Paper-04-2016.pdf

See Also

symextreme, scan_curve, scan_noisy_curve

Examples

```
## Lets create a convex/concave curve and classify it:
f=function(x){5+5*tanh(x-5)}
x=seq(0,8,0.01)
y=f(x)
plot(x,y,pch=19,cex=0.2)
cc=classify_curve(x,y)
cc$ctype
## [1] "convex_concave"
cc$index
## [1] 0
## Use 'index':
ede(x,y,cc$index)
##      j1  j2 chi
## EDE 369 633   5
## Lets create an 'almost convex' curve and see it:
f=function(x){-x^3+5*x-6}
x=seq(-2.5,1.5,0.01)
y=f(x)
plot(x,y,pch=19,cex=0.2)
cc=classify_curve(x,y)
cc$totalconvexity
## [1] "convex"
## Check for existence of a maximum for a noisy curve:
f=function(x){100-(x-5)^2}
x=seq(0,12,by=0.01);
r=1.5;y=f(x)+runif(length(x),-r,r)
plot(x,y,pch=19,cex=0.2)
cc=classify_curve(x,y)
cc$ismax
## [1] TRUE
```

`extremexi` *Function to Find the Extreme Point of a Planar Curve*

Description

It takes as input the x, y numeric vectors, the indices for the range to be searched plus some other options and finds the extreme point for that interval, while it plots data, Taylor polynomial and the computed $|a_1|$ coefficients.

Usage

```
extremexi(x, y, i1, i2, nt, alpha = 5, xlb = "x", ylb = "y", xnd = 3,  ynd = 3,
plots = TRUE, plotpdf = FALSE, doparallel=FALSE)
```

Arguments

`x`	A numeric vector for the independent variable
`y`	A numeric vector for the dependent variable
`i1`	The first index for choosing a specific interval $[a, b] = [x_{i1}, x_{i2}]$
`i2`	The second index for choosing a specific interval $[a, b] = [x_{i1}, x_{i2}]$
`nt`	The degree of the Taylor polynomial that will be fitted to the data
`alpha`	The level of statistical significance for the confidence intervals of coefficients $a_0, a_1, ..., a_{nt-1}$ (default value = 5)
`xlb`	A label for the x-variable (default value = "x")
`ylb`	A label for the y-variable (default value = "y")
`xnd`	The number of digits for plotting the x-axis (default value = 3)
`ynd`	The number of digits for plotting the y-axis (default value = 3)
`plots`	If plots=TRUE then a plot is created on default monitor (default value = TRUE)
`plotpdf`	If plotpdf=TRUE then a pdf plot is created and stored on working directory (default value = FALSE)
`doparallel`	If doparallel=TRUE then parallel computing is applied, based on the available workers of current machine (default value = FALSE)

Details

The point x_i which makes the relevant $|a_1|$ minimum is the estimation for the function's extreme point at the interval $[x_{i1}, x_{i2}]$.

Value

It returns an environment with two components:

`an`	a matrix with 3 columns: lower, upper bound of confidence interval and middle value for each coefficient an
`fextr`	a list with 2 members: the position i and the value of the estimated extreme point $\rho = x_i$

Warnings

When you are using RStudio it is necessary to leave enough space for the plot window in order for the plots to appear normally. The data should come from a function at least $C^{(1)}$ in order to be able to find an extreme point, if exists.

Author(s)

Demetris T. Christopoulos

References

Demetris T. Christopoulos (2014). Roots, extrema and inflection points by using a proper Taylor regression procedure. *SSRN*. doi:10.2139/ssrn.2521403

Examples

```
#Load data:
#
data(xydat)
#
#Extract x and y variables:
#
x=xydat$x;y=xydat$y
#
#Find extreme point, plot results, print Taylor coefficients and rho estimation:
#
c<-extremexi(x,y,1,length(x),5,5,plots=TRUE);c$an;c$fextr;
#
#Find multiple extrema.
#Let's create some data:
#
f=function(x){3*cos(x-5)};xa=0.;xb=9;
set.seed(12345);x=sort(runif(101,xa,xb));r=0.1;y=f(x)+2*r*(runif(length(x))-0.5);plot(x,y)
#
#The first extreme point is
c1<-extremexi(x,y,1,40,5,5,plots=TRUE);c1$an;c1$fextr;
#          2.5 %      97.5 %           an
# a0 -3.02708631 -2.94592364 -2.986504975
# a1  0.07660314  0.24706531  0.161834227
# a2  1.42127770  1.58580632  1.503542012
# a3 -0.09037154  0.10377241  0.006700434
# a4 -0.14788899 -0.08719428 -0.117541632
# a5 -0.03822416  0.01425066 -0.011986748
# [1] 22.000000  1.917229
#Compare it with the actual rho_1=1.858407346
#
#The second extreme point is
c2<-extremexi(x,y,50,80,5,5,plots=TRUE);c2$an;c2$fextr;
#          2.5 %       97.5 %         an
# a0  2.89779980  3.064703163  2.9812515
# a1  0.27288720  0.541496278  0.4071917
# a2 -1.81454401 -0.677932480 -1.2462382
```

```
# a3 -1.76290384  0.216201349 -0.7733512
# a4  0.02548354  1.269671304  0.6475774
# a5 -0.25156866  0.007565154 -0.1220018
# [1] 7.000000 4.896521
#You have to compare it with the actual value of rho_2=5.0
#
#Finally the third extreme point is
c3<-extremexi(x,y,80,length(x),5,5,plots=TRUE);c3$an;c3$fextr;
#         2.5 %     97.5 %         an
# a0 -3.0637461 -2.9218614 -2.9928037
# a1 -0.2381605  0.2615635  0.0117015
# a2  0.7860259  2.0105383  1.3982821
# a3 -1.4187417  0.7472155 -0.3357631
# a4 -0.7943208  1.0876143  0.1466468
# a5 -0.6677733  1.7628833  0.5475550
# [1] 11.000000  8.137392
#You have to compare it with the actual value of rho_3=8.141592654
```

`findextreme` *Implementation of Integration Extreme Finding Estimator (IEFE) for extreme points identification*

Description

Given a noisy or not planar curve as a set of discrete $\{(x_i, y_i), i = 1, 2, \ldots n\}$ points we use Integration Extreme Finding Estimator (IEFE) algorithm as it is described at [1] in ordfer to find the extreme point of it.

Usage

```
findextreme(x, y, parallel = FALSE, silent = TRUE, tryfast = FALSE)
```

Arguments

`x`	A numeric vector for the independent variable without missing values
`y`	A numeric vector for the dependent variable without missing values
`parallel`	Logical input, if TRUE then parallel processing will be used (default=FALSE)
`silent`	Logical input, if TRUE then no details will be printed out during code execution (default=TRUE)
`tryfast`	Logical input, if TRUE then instead 'BEDE' will be used from IEFE algorithm instead of BESE (default=FALSE)

Details

The parallel=TRUE option must be used if length(x)>20000. The tryfast=TRUE can be used for big data sets, but BEDE is not so accuracy as BESE, so use it with caution.

Value

A named vector with next components is returned:

1. x1 the left endpoint of the final interval of BESE or BEDE iterations
2. x2 the right endpoint of the final interval of BESE or BEDE iterations
3. chi the estimation of extreme as x-abscissa
4. ychi the estimation of extreme as y-abscissa taken from the interpolation polynomial of 2nd degree for the data points (x1,y1), (x2,y2), (chi,ychi)

Note

The 'yvalue' at output vector is an interpolation approxiamtion for the y-value of unknown function at its extreme point 'chi' and does not mean that it will be certainly accurate. Thta is the truth if underlying function can be well approximated by low order polynomials.

Author(s)

Demetris T. Christopoulos

References

[1]Demetris T. Christopoulos (2019). New methods for computing extremes and roots of a planar curve: introducing Noisy Numerical Analysis (2019). *ResearchGate*. doi:10.13140/RG.2.2.17158.32324

See Also

symextreme, findmaxtulip, findmaxbell

Examples

```
## Legendre polynomial 5th order
## True extreme point p=0.2852315165, y=0.3466277
f=function(x){(63/8)*x^5-(35/4)*x^3+(15/8)*x}
x=seq(0,0.7,0.001);y=f(x)
plot(x,y,pch=19,cex=0.5)
a=findextreme(x,y)
a
##        x1        x2       chi    yvalue
## 0.2840000 0.2860000 0.2850000 0.3466274
sol=a['chi']
abline(h=0)
abline(v=sol)
abline(v=a[1:2],lty=2)
abline(h=f(sol),lty=2)
points(sol,f(sol),pch=17,cex=2)
#
## The same function with noise from U(-0.05,0.05)
set.seed(2019-07-26);r=0.05;y=f(x)+runif(length(x),-r,r)
plot(x,y,pch=19,cex=0.5)
```

```
a=findextreme(x,y)
a
##        x1        x2       chi    yvalue
## 0.2890000 0.2910000 0.2900000 0.3895484
sol=a['chi']
abline(h=0)
abline(v=sol)
abline(v=a[1:2],lty=2)
abline(h=f(sol),lty=2)
points(sol,f(sol),pch=17,cex=2)
#
```

`findmaxbell` *Implementation of Bell Extreme Finding Estimator (BEFE) algorithm for a planar curve*

Description

For a curve that can be classified as 'bell' a fast computation of its maximum is performed by applying Bell Extreme Finding Estimator (BEFE) algorithm of [1].

Usage

```
findmaxbell(x, y, concave = TRUE)
```

Arguments

x	A numeric vector for the independent variable without missing values
y	A numeric vector for the dependent variable without missing values
concave	Logical input, if TRUE then curve is supposed to have a maximum (default=TRUE)

Details

If we want to compute minimum we just set concave=FALSE and proceed.

Value

A named vector with next components is returned:

1. the index of x-left
2. j2 the index of x-right
3. chi the estimation of extreme as x-abscissa

Note

Please use function `classify_curve` if you have not visual inspection in order to find the extreme type. Do not use that function if curve shape is not 'bell', use either `symextreme` or `findextreme`.

Author(s)

Demetris T. Christopoulos

References

[1]Demetris T. Christopoulos (2019). New methods for computing extremes and roots of a planar curve: introducing Noisy Numerical Analysis (2019). *ResearchGate*. doi:10.13140/RG.2.2.17158.32324

See Also

classify_curve, symextreme, findmaxtulip, findextreme

Examples

```
#
f=function(x){1/(1+x^2)}
x=seq(-2,2.0,by=0.01);y=f(x)
plot(x,y,pch=19,cex=0.5)
cc=classify_curve(x,y)
cc$shapetype
## 1] "bell"
a1<-findmaxbell(x,y)
a1
## j1           j2          chi
## 1.770000e+02 2.250000e+02 1.110223e-16
abline(v=a1['chi'])
abline(v=x[a1[1:2]],lty=2);abline(h=y[a1[1:2]],lty=2)
points(x[a1[1]:a1[2]],y[a1[1]:a1[2]],pch=19,cex=0.5,col='blue')
#
## Same curve with noise from U(-0.05,0.05)
set.seed(2019-07-26);r=0.05;y=f(x)+runif(length(x),-r,r)
plot(x,y,pch=19,cex=0.5)
cc=classify_curve(x,y)
cc$shapetype
## 1] "bell"
a1<-findmaxbell(x,y)
a1
##     j1     j2    chi
## 169.00 229.00  -0.02
abline(v=a1['chi'])
abline(v=x[a1[1:2]],lty=2);abline(h=y[a1[1:2]],lty=2)
points(x[a1[1]:a1[2]],y[a1[1]:a1[2]],pch=19,cex=0.5,col='blue')
#
```

`findmaxtulip` *Implementation of Tulip Extreme Finding Estimator (TEFE) algorithm for a planar curve*

Description

For a curve that can be classified as 'tulip' a fast computation of its maximum is performed by applying Tulip Extreme Finding Estimator (TEFE) algorithm of [1].

Usage

```
findmaxtulip(x, y, concave = TRUE)
```

Arguments

x	A numeric vector for the independent variable without missing values
y	A numeric vector for the dependent variable without missing values
concave	Logical input, if TRUE then curve is supposed to have a maximum (default=TRUE)

Details

If we want to compute minimum we just set concave=FALSE and proceed.

Value

A named vector with next components is returned:

1. j1 the index of x vextor that is the left endpoint of final symmetrization interval
2. j1 the index of x vextor that is the right endpoint of final symmetrization interval
3. chi the estimation of extreme as x-abscissa

Note

Please use function `classify_curve` if you have not visual inspection in order to find the extreme type. Do not use that function if curve shape is not 'tulip', use either `symextreme` or `findextreme`.

Author(s)

Demetris T. Christopoulos

References

[1]Demetris T. Christopoulos (2019). New methods for computing extremes and roots of a planar curve: introducing Noisy Numerical Analysis (2019). *ResearchGate*. doi:10.13140/RG.2.2.17158.32324

See Also

`classify_curve`, `symextreme`, `findmaxbell`, `findextreme`

Examples

```
#
f=function(x){100-(x-5)^2}
x=seq(0,12,by=0.01);y=f(x)
plot(x,y,pch=19,cex=0.5)
cc=classify_curve(x,y)
cc$shapetype
## 1] "tulip"
a<-findmaxtulip(x,y)
a
## j1   j2  chi
##  1 1001    5
abline(v=a['chi'])
abline(v=x[a[1:2]],lty=2);abline(h=y[a[1:2]],lty=2)
points(x[a[1]:a[2]],y[a[1]:a[2]],pch=19,cex=0.5,col='blue')
#
## Same curve with noise from U(-1.5,1.5)
set.seed(2019-07-26);r=1.5;y=f(x)+runif(length(x),-r,r)
plot(x,y,pch=19,cex=0.5)
cc=classify_curve(x,y)
cc$shapetype
## 1] "tulip"
plot(x,y,pch=19,cex=0.5)
a<-findmaxtulip(x,y)
a
##    j1       j2      chi
## 1.000 1002.000    5.005
abline(v=a['chi'])
abline(v=x[a[1:2]],lty=2);abline(h=y[a[1:2]],lty=2)
points(x[a[1]:a[2]],y[a[1]:a[2]],pch=19,cex=0.5,col='blue')
#
```

`findroot` *Find the root for a planar curve by using Integration Root Finding Estimator (IEFE) algorithm*

Description

Given a noisy or not planar curve as a set of discrete $\{(x_i, y_i), i = 1, 2, \ldots n\}$ points we use Integration Root Finding Estimator (IRFE) algorithm as it is described at [1] in ordfer to find the root of it.

Usage

```
findroot(x, y, parallel = FALSE, silent = TRUE, tryfast = FALSE)
```

Arguments

x	A numeric vector for the independent variable without missing values
y	A numeric vector for the dependent variable without missing values
parallel	Logical input, if TRUE then parallel processing will be used (default=FALSE)
silent	Logical input, if TRUE then no details will be printed out during code execution (default=TRUE)
tryfast	Logical input, if TRUE then instead 'BEDE' will be used from IEFE algorithm instead of BESE (default=FALSE)

Details

The parallel=TRUE otpion must be used if length(x)>20000. The tryfast=TRUE can be used for big data sets, but BEDE is not so accuracy as BESE, so use it with caution.

Value

A named vector with next components is returned:

1. the left endpoint of the final interval of BESE or BEDE iterations
2. x2 the right endpoint of the final interval of BESE or BEDE iterations
3. chi the estimation of extreme as x-abscissa
4. chi the estimation of extreme as y-abscissa taken from the interpolation polynomial of 2nd degree for the data points (x1,y1), (x2,y2), (chi,ychi)

Note

The 'yvalue' at output vector is an interpolation approximation for the y-value of unknown function at its extreme point 'chi' and does not mean that it will be certainly accurate. Thta is the truth if underlying function can be well approximated by low order polynomials.

Author(s)

Demetris T. Christopoulos

References

[1]Demetris T. Christopoulos (2019). New methods for computing extremes and roots of a planar curve: introducing Noisy Numerical Analysis (2019). *ResearchGate*. doi:10.13140/RG.2.2.17158.32324

See Also

scan_curve, scan_noisy_curve

Examples

```
#
## Legendre polynomial 5th order
f=function(x){(63/8)*x^5-(35/4)*x^3+(15/8)*x}
x=seq(0.2,0.8,0.001);y=f(x);ya=abs(y)
plot(x,y,pch=19,cex=0.5,ylim=c(min(y),max(ya)))
abline(h=0);
lines(x,ya,lwd=4,col='blue')
rt=findroot(x,y)
rt
##           x1            x2           chi        yvalue
## 5.370000e-01  5.400000e-01  5.385000e-01 -7.442574e-05
abline(v=rt['chi'])
abline(v=rt[1:2],lty=2);abline(h=rt['yvalue'],lty=2)
points(rt[3],rt[4],pch=17,col='blue',cex=2)
#
## Same curve but with noise from U(-0.5,0.5)
#
set.seed(2019-07-24);r=0.05;y=f(x)+runif(length(x),-r,r)
ya=abs(y)
plot(x,y,pch=19,cex=0.5,ylim=c(min(y),max(ya)))
abline(h=0)
points(x,ya,pch=19,cex=0.5,col='blue')
rt=findroot(x,y)
rt
##         x1          x2         chi      yvalue
## 0.53400000  0.53700000  0.53550000 -0.01762159
abline(v=rt['chi'])
abline(v=rt[1:2],lty=2);abline(h=rt['yvalue'],lty=2)
points(rt[3],rt[4],pch=17,col='blue',cex=2)
#
```

`inflexi` *Function to Find the Inflection Point of a Planar Curve*

Description

It takes as input the x, y numeric vectors, the indices for the range to be searched plus some other options and finds the inflection point for that interval, while it plots data, Taylor polynomial and and the computed $|a_2|$ coefficients.

Usage

```
inflexi(x, y, i1, i2, nt, alpha = 5, xlb = "x", ylb = "y", xnd = 3,  ynd = 3,
plots = TRUE, plotpdf = FALSE, doparallel=FALSE)
```

Arguments

x	A numeric vector for the independent variable
y	A numeric vector for the dependent variable
i1	The first index for choosing a specific interval $[a, b] = [x_{i1}, x_{i2}]$
i2	The second index for choosing a specific interval $[a, b] = [x_{i1}, x_{i2}]$
nt	The degree of the Taylor polynomial that will be fitted to the data
alpha	The level of statistical significance for the confidence intervals of coefficients $a_0, a_1, ..., a_{nt-1}$ (default value = 5)
xlb	A label for the x-variable (default value = "x")
ylb	A label for the y-variable (default value = "y")
xnd	The number of digits for plotting the x-axis (default value = 3)
ynd	The number of digits for plotting the y-axis (default value = 3)
plots	If plots=TRUE then a plot is created on default monitor (default value = TRUE)
plotpdf	If plotpdf=TRUE then a pdf plot is created and stored on working directory (default value = FALSE)
doparallel	If doparallel=TRUE then parallel computing is applied, based on the available workers of current machine (default value = FALSE)

Details

The point x_i which makes the relevant $|a_2|$ minimum is the estimation for the function's inflection point at the interval $[x_{i1}, x_{i2}]$.

Value

It returns an environment with two components:

an	a matrix with 3 columns: lower, upper bound of confidence interval and middle value for each coefficient an
fextr	a list with 2 members: the position i and the value of the estimated inflection point $\rho = x_i$

Warnings

When you are using RStudio it is necessary to leave enough space for the plot window in order for the plots to appear normally. The data should come from a function at least $C^{(2)}$ in order to be able to find an inflection point, if exists.

Author(s)

Demetris T. Christopoulos

References

Demetris T. Christopoulos (2014). Roots, extrema and inflection points by using a proper Taylor regression procedure. *SSRN*. doi:10.2139/ssrn.2521403

Examples

```
#Load data:
#
data(xydat)
#
#Extract x and y variables:
#
x=xydat$x;y=xydat$y
#
#Find inflection point, plot results, print Taylor coefficients and rho estimation:
#
d<-inflexi(x,y,1,length(x),5,5,plots=TRUE);d$an;d$finfl;
#
#Find multiple inflection points.
#Let's create some data:
#
f=function(x){3*cos(x-5)};xa=0.;xb=9;
set.seed(12345);x=sort(runif(101,xa,xb));r=0.1;y=f(x)+2*r*(runif(length(x))-0.5);plot(x,y)
#
#The first inflection point is
d1<-inflexi(x,y,20,50,5,5,plots=TRUE);d1$an;d1$finfl;
#         2.5 %     97.5 %           an
# a0  0.1483905  0.2377617  0.193076089
# a1  2.9024852  3.0936024  2.998043835
# a2 -0.2053120  0.2220390  0.008363525
# a3 -0.5845597 -0.3426017 -0.463580702
# a4 -0.2431038  0.1136244 -0.064739689
# a5 -0.0893246  0.0687848 -0.010269897
# [1] 19.000000  3.493296
#Compare it with the actual rho_1=3.429203673
#
#The second inflection point is
# d2<-inflexi(x,y,50,length(x),5,5,plots=TRUE);d2$an;d2$finfl;
#           2.5 %       97.5 %            an
# a0 -0.000875677  0.057156356  0.0281403394
# a1 -3.058363342 -2.942026810 -3.0001950762
# a2 -0.056224101  0.044135857 -0.0060441222
# a3  0.433135897  0.528446241  0.4807910691
# a4 -0.011774733  0.012002414  0.0001138404
# a5 -0.026899286 -0.009520899 -0.0182100925
# [1] 23.000000  6.567948
#You have to compare it with the actual value of rho_2=6.570796327
```

---

rootexinf *Function to Find the Root, Extreme and Inflection of a Planar Curve*

Description

It takes as input the x, y numeric vectors, the indices for the range to be searched plus some other options and finds the root, extreme and inflection for that interval, while it plots data, Taylor polynomial and and the computed $|a_0|$, $|a_1|$, $|a_2|$ coefficients.

Usage

```
rootexinf(x, y, i1, i2, nt, alpha = 5, xlb = "x", ylb = "y", xnd = 3,  ynd = 3,
plots = TRUE, plotpdf = FALSE, doparallel=FALSE)
```

Arguments

x	A numeric vector for the independent variable
y	A numeric vector for the dependent variable
i1	The first index for choosing a specific interval $[a, b] = [x_{i1}, x_{i2}]$
i2	The second index for choosing a specific interval $[a, b] = [x_{i1}, x_{i2}]$
nt	The degree of the Taylor polynomial that will be fitted to the data
alpha	The level of statistical significance for the confidence intervals of coefficients $a_0, a_1, ..., a_{nt-1}$ (default value = 5)
xlb	A label for the x-variable (default value = "x")
ylb	A label for the y-variable (default value = "y")
xnd	The number of digits for plotting the x-axis (default value = 3)
ynd	The number of digits for plotting the y-axis (default value = 3)
plots	If plots=TRUE then a plot is created on default monitor (default value = TRUE)
plotpdf	If plotpdf=TRUE then a pdf plot is created and stored on working directory (default value = FALSE)
doparallel	If doparallel=TRUE then parallel computing is applied, based on the available workers of current machine (default value = FALSE)

Details

The points x_i that make the relevant $|a_0|$, $|a_1|$, $|a_2|$ minimum are the estimations for the function's root, etreme and inflection point at the interval $[x_{i1}, x_{i2}]$.

Value

It returns an environment with four components:

an0	a matrix with 3 columns: lower, upper bound of confidence interval and middle value for each coefficient a_n at the best choice in root searching
an1	a matrix with 3 columns: lower, upper bound of confidence interval and middle value for each coefficient a_n at the best choice in extreme searching
an2	a matrix with 3 columns: lower, upper bound of confidence interval and middle value for each coefficient a_n at the best choice in inflection searching
frexinf	a 3 x 3 matrix: for each row (root, extreme, inflection) the position i and the value of the estimated root, extreme and inflection $\rho = x_i$

Warnings

When you are using RStudio it is necessary to leave enough space for the plot window in order for the plots to appear normally. The data should come from a function at least $C^{(2)}$ in order to find the root, extreme and inflection point, provided those points exist.

Author(s)

Demetris T. Christopoulos

References

Demetris T. Christopoulos (2014). Roots, extrema and inflection points by using a proper Taylor regression procedure. *SSRN*. doi:10.2139/ssrn.2521403

Examples

```
#Load data:
#Let's create some data:
f=function(x){3*cos(x-5)+1.5};xa=1.;xb=5;
set.seed(12345);x=sort(runif(5001,xa,xb));
r=0.1;y=f(x)+2*r*(runif(length(x))-0.5);plot(x,y);abline(h=0)
#a<-rootexinf(x,y,1,length(x),5,plotpdf = TRUE,doparallel = TRUE);a$an0;a$an1;a$an2;a$frexinf;
# Available workers are 12
# Time difference of 13.02153 secs
# File 'root_extreme_inflection_plot.pdf' has been created
#           2.5 %       97.5 %          an0
# a0 -0.004165735  0.001838624 -0.001163555
# a1  2.588990973  2.600915136  2.594953055
# a2  0.731456294  0.741262772  0.736359533
# a3 -0.435591038 -0.423837041 -0.429714040
# a4 -0.052926049 -0.050039975 -0.051483012
# a5  0.017915715  0.020538155  0.019226935
#           2.5 %       97.5 %           an1
# a0 -1.507117843 -1.500375848 -1.5037468451
# a1 -0.008343275  0.007916087 -0.0002135941
# a2  1.519432687  1.534103788  1.5267682378
# a3 -0.017663080  0.007780728 -0.0049411760
# a4 -0.159461025 -0.144303367 -0.1518821962
# a5  0.017915715  0.020538155  0.0192269354
#           2.5 %       97.5 %           an2
# a0  1.503394727  1.509925166  1.5066599466
# a1  2.985374546  2.995259021  2.9903167834
# a2 -0.009041165  0.005898692 -0.0015712367
# a3 -0.489107253 -0.480579585 -0.4848434187
# a4 -0.003885327  0.002364758 -0.0007602842
# a5  0.017915715  0.020538155  0.0192269354
# index    value
# root        2364 2.903791
# extreme     1057 1.859431
# inflection  3038 3.431413
# You have to compare with the exact values
# root=2.905604898
# extreme=1.858407346
# inflection=3.429203673
```

rootxi *Function to Find the Root of a Planar Curve*

Description

It takes as input the x, y numeric vectors, the indices for the range to be searched plus some other options and finds the root for that interval, while it plots data, Taylor polynomial and and the computed $|a_0|$ coefficients.

Usage

```
rootxi(x, y, i1, i2, nt, alpha = 5, xlb = "x", ylb = "y", xnd = 3,  ynd = 3,
plots = TRUE, plotpdf = FALSE, doparallel=FALSE)
```

Arguments

x	A numeric vector for the independent variable
y	A numeric vector for the dependent variable
i1	The first index for choosing a specific interval $[a, b] = [x_{i1}, x_{i2}]$
i2	The second index for choosing a specific interval $[a, b] = [x_{i1}, x_{i2}]$
nt	The degree of the Taylor polynomial that will be fitted to the data
alpha	The level of statistical significance for the confidence intervals of coefficients $a_0, a_1, ..., a_{nt-1}$ (default value = 5)
xlb	A label for the x-variable (default value = "x")
ylb	A label for the y-variable (default value = "y")
xnd	The number of digits for plotting the x-axis (default value = 3)
ynd	The number of digits for plotting the y-axis (default value = 3)
plots	If plots=TRUE then a plot is created on default monitor (default value = TRUE)
plotpdf	If plotpdf=TRUE then a pdf plot is created and stored on working directory (default value = FALSE)
doparallel	If doparallel=TRUE then parallel computing is applied, based on the available workers of current machine (default value = FALSE)

Details

The point x_i which makes the relevant $|a_0|$ minimum is the estimation for the function's root at the interval $[x_{i1}, x_{i2}]$.

Value

It returns an environment with two components:

an	a matrix with 3 columns: lower, upper bound of confidence interval and middle value for each coefficient a_n
froot	a list with 2 members: the position i and the value of the estimated root $\rho = x_i$

Warnings

When you are using RStudio it is necessary to leave enough space for the plot window in order for the plots to appear normally. The data should come from a function at least $C^{(0)}$ in order to find the root, provided that such a root exists.

Author(s)

Demetris T. Christopoulos

References

Demetris T. Christopoulos (2014). Roots, extrema and inflection points by using a proper Taylor regression procedure. *SSRN*. doi:10.2139/ssrn.2521403

Examples

```
#Load data:
#
data(xydat)
#
#Extract x and y variables:
#
x=xydat$x;y=xydat$y
#
#Find root, plot results, print Taylor coefficients and rho estimation:
#
b<-rootxi(x,y,1,length(x),5,5,plots=TRUE);b$an;b$froot;
#
#Find multiple roots.
#Let's create some data:
#
f=function(x){3*cos(x-5)};xa=0.;xb=9;
set.seed(12345);x=sort(runif(101,xa,xb));r=0.1;y=f(x)+2*r*(runif(length(x))-0.5);plot(x,y)
#
#The first root is
#
b1<-rootxi(x,y,1,20,5,5,plots=TRUE);b1$an;b1$froot;
#          2.5 %      97.5 %          an
# a0 -0.09380972  0.03295954 -0.03042509
# a1 -3.63025679 -2.89908741 -3.26467210
# a2 -0.90435090  0.80658742 -0.04888174
# a3 -1.27911360  6.88168053  2.80128346
# a4 -8.77763032  2.51983279 -3.12889877
# a5 -1.10798564  3.38419904  1.13810670
# [1] 5.0000000 0.3108189
#Compare it with the actual rho_1=0.2876110196
#
#The second root is
#
b2<-rootxi(x,y,20,50,5,5,plots=TRUE);b2$an;b2$froot;
#         2.5 %     97.5 %          an
# a0  0.1483905  0.2377617  0.193076089
```

```
# a1  2.9024852  3.0936024  2.998043835
# a2 -0.2053120  0.2220390  0.008363525
# a3 -0.5845597 -0.3426017 -0.463580702
# a4 -0.2431038  0.1136244 -0.064739689
# a5 -0.0893246  0.0687848 -0.010269897
# [1] 19.000000  3.493296
#You have to compare it with the actual value of rho_2=3.429203673
#
#Finally the third root is
b3<-rootxi(x,y,50,90,5,5,plots=TRUE);b3$an;b3$froot;
#           2.5 %       97.5 %            an
# a0 -0.002269152  0.058784414  0.0282576308
# a1 -3.090980046 -2.938875341 -3.0149276930
# a2 -0.089893659  0.075094637 -0.0073995112
# a3  0.403040978  0.591836654  0.4974388159
# a4 -0.035442477  0.037165754  0.0008616385
# a5 -0.048414145  0.005815106 -0.0212995192
# [1] 23.000000  6.567948
#You have to compare it with the actual value of rho_3=6.570796327
```

---

scan_curve *Finds roots, extrema and inflections for a planar not noisy curve*

Description

Given a not noisy planar curve as a set of discrete $\{(x_i, y_i), i = 1, 2, \ldots n\}$ points we use Integration Root Finding Estimator (IRFE), Integration Extreme Finding Estimator (IEFE) and BESE in order to find all roots and the enclosed between them extrema and infletion points. Estimators are defined and described at [1] and [2], [3].

Usage

```
scan_curve(x, y, findroots = TRUE, findextremes = TRUE,
   findinflections = TRUE, silent = FALSE, plots = TRUE)
```

Arguments

x	A numeric vector for the independent variable without missing values
y	A numeric vector for the dependent variable without missing values
findroots	Logical input, if TRUE then find all existing roots (default=TRUE)
findextremes	Logical input, if TRUE then find all existing extrema between the found roots (default=TRUE)
findinflections	Logical input, if TRUE then find all existing inflection points between the found roots (default=TRUE)
silent	Logical input, if TRUE then no details will be printed out during code execution (default=TRUE)
plots	Logical input, if TRUE then a plot with all roots, extrema and inflctions will be created (default=TRUE)

Details

The function first makes a study for the curve based on the interval classification done by absolute y-values. Then it uses an extensive if-else environment in order to cover all possible cases without error breaks.

Value

A list with next memebers is returned:

1. study a data frame with all intervals that curve can be divided and next columns:
   - x the root estimation taken from absolute value
   - y the y-value for root estimation x
   - y the absolute y-value for root estimation x
   - zero if TRUE, then all values of transformed yi's are zero for current interval
   - dif monotonicty index, if it is +1 (-1) then curve is increasing (decreasing)at root x
   - ja the index of x vector for the left endpoint of current interval
   - jb the index of x vector for the right endpoint of current interval
   - root_monotonicity monotonicty characterization for current root x
   - extremity extremity characterization for current interval
   - sigmoidicity sigmoidicity characterization for current interval
2. roots a data frame with all roots and next columns:
   - x1 the left endpoint of the final interval of BESE iterations
   - x2 the right endpoint of the final interval of BESE iterations
   - chi the estimation of root as x-abscissa
   - chi the estimation of root as y-abscissa taken from the interpolation polynomial of 2nd degree for the data points (x1,y1), (x2,y2), (chi,ychi)
3. extremes a data frame with all extremes between roots and next columns:
   - x1 the left endpoint of the final interval of BESE iterations
   - x2 the right endpoint of the final interval of BESE iterations
   - chi the estimation of extreme as x-abscissa
   - chi the estimation of extreme as y-abscissa taken from the interpolation polynomial of 2nd degree for the data points (x1,y1), (x2,y2), (chi,ychi)
4. inflections a data frame with all inflections between roots and next columns:
   - x1 the left endpoint of the final interval of BESE iterations
   - x2 the right endpoint of the final interval of BESE iterations
   - chi the estimation of inflection as x-abscissa
   - chi the estimation of inflection as y-abscissa taken from the interpolation polynomial of 2nd degree for the data points (x1,y1), (x2,y2), (chi,ychi)

Note

A heuristic is first used for determination of noise: if curve is classified as a noisy one, then only 'study' and a rough plot are available.

Author(s)

Demetris T. Christopoulos

References

[1]Demetris T. Christopoulos (2019). New methods for computing extremes and roots of a planar curve: introducing Noisy Numerical Analysis (2019). *ResearchGate*. doi:10.13140/RG.2.2.17158.32324

[2]Demetris T. Christopoulos (2014). Developing methods for identifying the inflection point of a convex/concave curve. arXiv:1206.5478v2 [math.NA]. doi:10.48550/arXiv.1206.5478

[3]Demetris T. Christopoulos (2016). On the efficient identification of an inflection point.International Journal of Mathematics and Scientific Computing, (ISSN: 2231-5330), vol. 6(1). https://demovtu.veltech.edu.in/wp-content/uploads/2016/04/Paper-04-2016.pdf

See Also

scan_noisy_curve, classify_curve

Examples

```
#
## Legendre polynomial 5th order
f=function(x){(63/8)*x^5-(35/4)*x^3+(15/8)*x}
x=seq(-1,1,0.001);y=f(x)
plot(x,y,pch=19,cex=0.5)
abline(h=0)
rall=scan_curve(x,y)
rall$study
rall$roots
##          x1     x2           chi        yvalue
## [1,] -0.907 -0.905 -9.060000e-01  1.234476e-03
## [2,] -0.540 -0.537 -5.385000e-01  7.447856e-05
## [3,] -0.001  0.001  5.551115e-17  1.040844e-16
## [4,]  0.537  0.540  5.385000e-01 -7.444324e-05
## [5,]  0.905  0.907  9.060000e-01 -1.234476e-03
rall$extremes
##          x1     x2    chi     yvalue
## [1,] -0.766 -0.764 -0.765  0.4196969
## [2,] -0.286 -0.284 -0.285 -0.3466274
## [3,]  0.284  0.286  0.285  0.3466274
## [4,]  0.764  0.766  0.765 -0.4196969
rall$inflections
##          x1     x2           chi        yvalue
## [1,] -0.579 -0.576 -5.775000e-01  9.659939e-02
## [2,] -0.001  0.001  5.551115e-17  1.040829e-16
## [3,]  0.576  0.579  5.775000e-01 -9.659935e-02
#
```

`scan_noisy_curve` *Finds roots, extrema and inflections for a planar noisy curve*

Description

Given a noisy planar curve as a set of discrete $\{(x_i, y_i), i = 1, 2, \ldots n\}$ points we use Integration Root Finding Estimator (IRFE), Integration Extreme Finding Estimator (IEFE) and BESE in order to find all roots and the enclosed between them extrema and inflction points. Estimators are defined and described at [1] and [2], [3].

Usage

```
scan_noisy_curve(x, y, noise = NULL, rootsoptim = TRUE, findextremes = TRUE,
  findinflections = TRUE, silent = FALSE, plots = TRUE)
```

Arguments

`x`	A numeric vector for the independent variable without missing values
`y`	A numeric vector for the dependent variable without missing values
`noise`	Either NULL (default value) or TRUE/FALSE input for suggesting that curve is indeed a noisy one
`rootsoptim`	Logical input, if TRUE then IRFE algorithm will also be used for roots (default=TRUE)
`findextremes`	Logical input, if TRUE then find all existing extrema between the found roots (default=TRUE)
`findinflections`	Logical input, if TRUE then find all existing inflection points between the found roots (default=TRUE)
`silent`	Logical input, if TRUE then no details will be printed out during code execution (default=TRUE)
`plots`	Logical input, if TRUE then a plot with all roots, extrema and inflctions will be created (default=TRUE)

Details

The function first uses `scan_curve` in norder to perform a first study. Then it uses an extesnsive if-else environment in order to cover all possible cases without error breaks.

Value

A list with next members is returned:

1. study a data frame with all intervals that curve can be divided and next columns:
   - j the index of x vector for the left endpoint of current interval
   - dj the number of x points of current interval

- interval logical, if TRUE then it is a unique convexity interval
- i1 the index of x vector for the left endpoint of current interval
- i2 the index of x vector for the right endpoint of current interval
- interval logical, if TRUE then a root exists inside current interval

2. roots_average a data frame with all averaged roots and next columns:
   - x1 the left endpoint of the averaged interval
   - x2 the right endpoint of the averaged interval
   - chi the estimation of root as x-abscissa
   - chi the estimation of root as y-abscissa taken from the interpolation polynomial of 2nd degree for a properly collected triad of (x[i],y[i]) points
3. roots_optim a data frame with all roots from IRFE and next columns:
   - x1 the left endpoint of the final interval of BESE iterations
   - x2 the right endpoint of the final interval of BESE iterations
   - chi the estimation of root as x-abscissa
   - chi the estimation of root as y-abscissa taken from the interpolation polynomial of 2nd degree for the data points (x1,y1), (x2,y2), (chi,ychi)
4. extremes a data frame with all extremes between roots and next columns:
   - x1 the left endpoint of the final interval of BESE iterations
   - x2 the right endpoint of the final interval of BESE iterations
   - chi the estimation of extreme as x-abscissa
   - chi the estimation of extreme as y-abscissa taken from the interpolation polynomial of 2nd degree for the data points (x1,y1), (x2,y2), (chi,ychi)
5. inflections a data frame with all inflections between roots and next columns:
   - x1 the left endpoint of the final interval of BESE iterations
   - x2 the right endpoint of the final interval of BESE iterations
   - chi the estimation of inflection as x-abscissa
   - chi the estimation of inflection as y-abscissa taken from the interpolation polynomial of 2nd degree for the data points (x1,y1), (x2,y2), (chi,ychi)

Note

If curve is not a noisy one, then `scan_curve` is used.

Author(s)

Demetris T. Christopoulos

References

[1]Demetris T. Christopoulos (2019). New methods for computing extremes and roots of a planar curve: introducing Noisy Numerical Analysis (2019). *ResearchGate*. doi:10.13140/RG.2.2.17158.32324

[2]Demetris T. Christopoulos (2014). Developing methods for identifying the inflection point of a convex/concave curve. arXiv:1206.5478v2 [math.NA]. doi:10.48550/arXiv.1206.5478

[3]Demetris T. Christopoulos (2016). On the efficient identification of an inflection point.International Journal of Mathematics and Scientific Computing, (ISSN: 2231-5330), vol. 6(1). https://demovtu.veltech.edu.in/wp-content/uploads/2016/04/Paper-04-2016.pdf

See Also

scan_curve, classify_curve

Examples

```
#
## Legendre polynomial 5th order
f=function(x){(63/8)*x^5-(35/4)*x^3+(15/8)*x}
x=seq(-1,1,0.001)
set.seed(2019-07-26);r=0.05;y=f(x)+runif(length(x),-r,r)
plot(x,y,pch=19,cex=0.5)
rn=scan_noisy_curve(x,y)
rn
## $study
##       j  dj interval   i1   i2  root
## 3    97 351     TRUE   97  448 FALSE
## 18  477 502     TRUE  477  979 FALSE
## 39 1021 505     TRUE 1021 1526 FALSE
## 54 1558 343     TRUE 1558 1901 FALSE
##
## $roots_average
##       x1     x2     chi       yvalue
## 1 -0.906 -0.904 -0.9050 -0.002342389
## 2 -0.553 -0.524 -0.5385  0.005003069
## 3 -0.022  0.020 -0.0010  0.003260937
## 4  0.525  0.557  0.5410 -0.007956680
## 5  0.900  0.911  0.9055 -0.008015683
##
## $roots_optim
##       x1     x2     chi       yvalue
## 1 -0.909 -0.901 -0.9050 -0.023334404
## 2 -0.531 -0.527 -0.5290  0.029256059
## 3  0.001  0.003  0.0020  0.001990572
## 4  0.530  0.565  0.5475  0.019616283
## 5  0.909  0.912  0.9105  0.009288338
##
## $extremes
##          x1     x2     chi     yvalue
## [1,] -0.773 -0.766 -0.7695  0.4102010
## [2,] -0.280 -0.274 -0.2770 -0.3804006
## [3,]  0.308  0.316  0.3120  0.3372764
## [4,]  0.741  0.744  0.7425 -0.4414494
##
## $inflections
##          x1     x2     chi       yvalue
## [1,] -0.772 -0.275 -0.5235 -0.076483193
## [2,] -0.275  0.281  0.0030 -0.007558037
```

```
## [3,]  0.301  0.776  0.5385  0.018958334
#
```

`symextreme` *A wrapper for using either Tulip Extreme Finding Estimator (TEFE) or Bell Extreme Finding Estimator (BEFE) algorithm for a symmetric extreme point*

Description

Function uses `classify_curve` and then either `findmaxbell` or `findmaxtulip` to proceed. See [1] for definitions and details of algorithms.

Usage

```
symextreme(x, y, concave = NULL, type = NULL)
```

Arguments

`x`	A numeric vector for the independent variable without missing values
`y`	A numeric vector for the dependent variable without missing values
`concave`	Logical input, if TRUE then curve is supposed to have a maximum (default=TRUE)
`type`	A character string inpute denoting the shape of the curve, either 'bell' or 'tulip' (default=NULL)

Details

This function is useful if we know that our curve has a symmetry around its extreme point but we cannot directly infer for the relevant shape.

Value

A list with next memebers is returned:

1. maximum logical, if TRUE then curve has a maximum
2. minimum logical, if TRUE then curve has a minimum
3. results a named vector with components:
   - j1 the index of x-left
   - j2 the index of x-right
   - chi the estimation of extreme as x-abscissa

Note

You can Use the 'type' input if you are sure for the shape of the curve.

Author(s)

Demetris T. Christopoulos

References

[1]Demetris T. Christopoulos (2019). New methods for computing extremes and roots of a planar curve: introducing Noisy Numerical Analysis (2019). *ResearchGate*. doi:10.13140/RG.2.2.17158.32324

See Also

classify_curve, findmaxtulip, findmaxbell, findextreme

Examples

```
#
## Bell curve
f=function(x){1/(1+x^2)}
x=seq(-2,2.0,by=0.01);y=f(x)
plot(x,y,pch=19,cex=0.5)
a=symextreme(x,y)
a
## $maximum
## [1] TRUE
##
## $minimum
## [1] FALSE
##
## $results
##           j1           j2          chi
## 1.770000e+02 2.250000e+02 1.110223e-16
abline(v=a$results['chi'])
#
## Tulip curve
f=function(x){100-(x-5)^2}
x=seq(0,12,by=0.01);y=f(x)
plot(x,y,pch=19,cex=0.5)
a=symextreme(x,y)
a
## $maximum
## [1] TRUE
##
## $minimum
## [1] FALSE
##
## $results
##     j1   j2  chi
## 1 1001    5
abline(v=a$results['chi'])
#
```

xydat *xydat*

Description

A dataset containing 61 xy-points

Usage

```
data("xydat")
```

Format

A data frame with 61 observations on the following 2 variables.

`x` a numeric vector

`y` a numeric vector

Examples

```
data(xydat)
```

Index

∗ **BEDE**
findextreme, 9
scan_curve, 23
scan_noisy_curve, 26
∗ **BEFE**
findmaxbell, 11
symextreme, 29
∗ **BESE**
findextreme, 9
∗ **IEFE**
findextreme, 9
scan_curve, 23
scan_noisy_curve, 26
∗ **IRFE**
findroot, 14
scan_curve, 23
scan_noisy_curve, 26
∗ **TEFE**
findmaxtulip, 12
symextreme, 29
∗ **asymmetry**
classify_curve, 5
∗ **bell**
classify_curve, 5
findmaxbell, 11
symextreme, 29
∗ **convexity**
classify_curve, 5
∗ **curve**
classify_curve, 5
∗ **datasets**
xydat, 31
∗ **extremexi**
extremexi, 7
inflexi, 16
RootsExtremaInflections-package, 2
∗ **extreme**
classify_curve, 5
∗ **inflection**
scan_curve, 23
scan_noisy_curve, 26
∗ **inflexi**
RootsExtremaInflections-package, 2
∗ **maximum**
findextreme, 9
findmaxbell, 11
findmaxtulip, 12
scan_curve, 23
scan_noisy_curve, 26
symextreme, 29
∗ **minimum**
findextreme, 9
findmaxbell, 11
findmaxtulip, 12
scan_curve, 23
scan_noisy_curve, 26
symextreme, 29
∗ **noise**
scan_noisy_curve, 26
∗ **rootexinf**
rootexinf, 18
∗ **rootxi**
RootsExtremaInflections-package, 2
rootxi, 21
∗ **root**
findroot, 14
scan_curve, 23
scan_noisy_curve, 26
∗ **symmetry**
symextreme, 29
∗ **tulip**
classify_curve, 5
findmaxtulip, 12
symextreme, 29

classify_curve, 5, *11–13*, *25*, *28–30*

extremexi, 7

findextreme, 9, *11–13*, *30*
findmaxbell, *10*, 11, *13*, *29, 30*
findmaxtulip, *10*, *12*, 12, *29, 30*
findroot, 14

inflexi, 16

rootexinf, 18
RootsExtremaInflections
(RootsExtremaInflections-package), 2
RootsExtremaInflections-package, 2
rootxi, 21

scan_curve, *6*, *15*, 23, *26–28*
scan_noisy_curve, *6*, *15*, *25*, 26
symextreme, *6*, *10–13*, 29

xydat, 31