Package 'itp'

January 10, 2026

Type Package

Title The Interpolate, Truncate, Project (ITP) Root-Finding Algorithm

Version 1.2.2

Date 2026-01-09

Description Implements the Interpolate, Truncate, Project (ITP) root-finding algorithm developed by Oliveira and Takahashi (2021) <doi:10.1145/3423597>. The user provides the function, from the real numbers to the real numbers, and an interval with the property that the values of the function at its endpoints have different signs. If the function is continuous over this interval then the ITP method estimates the value at which the function is equal to zero. If the function is discontinuous then a point of discontinuity at which the function changes sign may be found. The function can be supplied using either an R function or an external pointer to a C++ function. Tuning parameters of the ITP algorithm can be set by the user. Default values are set based on arguments in Oliveira and Takahashi (2021).

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.3.3

Imports Rcpp (>= 1.0.8)

LinkingTo Rcpp

Suggests knitr, rmarkdown, stats, testthat (>= 3.0.0)

VignetteBuilder knitr

URL https://paulnorthrop.github.io/itp/, https://github.com/paulnorthrop/itp

BugReports https://github.com/paulnorthrop/itp/issues

Config/testthat/edition 3

NeedsCompilation yes

Author Paul J. Northrop [aut, cre, cph]

Maintainer Paul J. Northrop <p.northrop@ucl.ac.uk>

Repository CRAN

Date/Publication 2026-01-10 08:00:15 UTC

Contents

`itp-package` *itp: The Interpolate, Truncate, Project (ITP) Root-Finding Algorithm*

Description

Implements the Interpolate, Truncate, Project (ITP) root-finding algorithm developed by Oliveira and Takahashi (2021). The user provides a function, from the real numbers to the real numbers, and an interval with the property that the values of the function at its endpoints have different signs. If the function is continuous over this interval then the ITP method estimates the value at which the function is equal to zero. If the function is discontinuous then a point of discontinuity at which the function changes sign may be found. The function can be supplied using either an R function or an external pointer to a C++ function. Tuning parameters of the ITP algorithm can be set by the user. Default values are set based on arguments in Oliveira and Takahashi (2021).

Details

The main function is `itp`. See the vignette Overview of the itp package, which can also be accessed using `vignette("itp-vignette", package = "itp")`.

Author(s)

Maintainer: Paul J. Northrop <p.northrop@ucl.ac.uk> [copyright holder]

References

Oliveira, I. F. D. and Takahashi, R. H. C. (2021). An Enhancement of the Bisection Method Average Performance Preserving Minmax Optimality, *ACM Transactions on Mathematical Software*, **47**(1), 1-24. doi:10.1145/3423597

See Also

`itp` for the ITP root-finding algorithm

`print.itp` and `plot.itp` for print and plot methods for objects of class `"itp"` returned from `itp`.

`xptr_create` and `xptr_eval` for creating external pointers to the C++ functions used as examples in this package and evaluating those functions.

`itp` *The ITP root-finding algorithm*

Description

Performs one-dimensional root-finding using the ITP algorithm of Oliveira and Takahashi (2021). The function `itp` searches an interval $[a, b]$ for a root (i.e., a zero) of the function `f` with respect to its first argument. Each iteration results in a bracketing interval for the root that is narrower than the previous interval. If the function is discontinuous then a point of discontinuity at which the function changes sign may be found.

Usage

```
itp(
  f,
  interval,
  ...,
  a = min(interval),
  b = max(interval),
  f.a = f(a, ...),
  f.b = f(b, ...),
  epsilon = 1e-10,
  k1 = 0.2/(b - a),
  k2 = 2,
  n0 = 1
)
```

Arguments

`f`	An R function or an external pointer to a C++ function. For the latter see the article Passing user-supplied C++ functions in the Rcpp Gallery. The function for which the root is sought.
`interval`	A numeric vector `c(a, b)` of length 2 containing the end points of the interval to be searched for the root. The function values at the end points must be of opposite signs.
`...`	Additional arguments to be passed to `f`.
`a, b`	An alternative way to set the lower and upper end points of the interval to be searched. The function values at these end points must be of opposite signs.
`f.a, f.b`	The values of `f(a)` and `f(b)`, respectively.

epsilon	A positive numeric scalar. The desired accuracy of the root. The algorithm continues until the width of the bracketing interval for the root is less than or equal to 2 * epsilon.
k1, k2, n0	Numeric scalars. The values of the tuning parameters κ_1, κ_2, n_0. See **Details**.

Details

Page 8 of Oliveira and Takahashi (2021) describes the ITP algorithm and the roles of the tuning parameters κ_1, κ_2 and n_0. The algorithm is described using pseudocode. The Wikipedia entry for the ITP method provides a summary. If the input function f is continuous over the interval [a, b] then the value of f evaluated at the estimated root is (approximately) equal to 0. If f is discontinuous over the interval [a, b] then the bracketing interval returned after convergence has the property that the signs of the function f at the end points of this interval are different and therefore the estimated root may be a point of discontinuity at which the sign of f changes.

The ITP method requires at most $n_{\max} = n_{1/2} + n_0$ iterations, where $n_{1/2}$ is the smallest integer not less than $\log_2((b-a)/2\epsilon)$. If $n_0 = 0$ then the ITP method will require no more iterations than the bisection method. Depending on the function f, setting a larger value for $n_0 = 0$, e.g. the default setting $n_0 = 1$ used by the itp function, may result in a smaller number of iterations.

The default values of the other tuning parameters (epsilon = 1e-10, k1 = 0.2 / (b - a), k2 = 2) are set based on arguments made in Oliveira and Takahashi (2021).

Value

An object (a list) of class "itp" containing the following components:

root	the location of the root, calculated as (a+b)/2, where [a, b] is the bracketing interval after convergence.
f.root	the value of the function evaluated at root.
iter	the number of iterations performed.
a, b	the end points of the bracketing interval [a, b] after convergence.
f.a, f.b	the values of function at a and b after convergence.
estim.prec	an approximate estimated precision for root, equal to the half the width of the final bracket for the root.

If the root occurs at one of the input endpoints a or b then iter = 0 and estim.prec = NA.

The returned object also has the attributes f (the input R function or pointer to a C++ function f), f_args (a list of additional arguments to f provided in ...), f_name (a function name extracted from as.character(substitute(f)) or the form of the R function if f was not named), used_c (a logical scalar: FALSE, if f is an R function and TRUE if f is a pointer to a C++ function) and input_a and input_b (the input values of a and b). These attributes are used in plot.itp to produce a plot of the function f over the interval (input_a, input_b).

References

Oliveira, I. F. D. and Takahashi, R. H. C. (2021). An Enhancement of the Bisection Method Average Performance Preserving Minmax Optimality, *ACM Transactions on Mathematical Software*, **47**(1), 1-24. doi:10.1145/3423597

See Also

print.itp and plot.itp for print and plot methods for objects of class "itp" returned from itp.

Examples

```
#### ----- The example used in the Wikipedia entry for the ITP method

# Supplying an R function
wiki <- function(x) x ^ 3 - x - 2
itp(wiki, c(1, 2), epsilon = 0.0005, k1 = 0.1, n0 = 1)
# The default setting (with k1 = 0.2) wins by 1 iteration
wres <- itp(wiki, c(1, 2), epsilon = 0.0005, n0 = 1)
wres
plot(wres)

# Supplying an external pointer to a C++ function
wiki_ptr <- xptr_create("wiki")
wres_c <- itp(f = wiki_ptr, c(1, 2), epsilon = 0.0005, k1 = 0.1)
wres_c
plot(wres_c)

#### ----- Some examples from Table 1 of Oliveira and Takahashi (2021)

### Well-behaved functions

# Lambert
lambert <- function(x) x * exp(x) - 1
itp(lambert, c(-1, 1))

# Trigonometric 1
# Supplying an R function
trig1 <- function(x, root) tan(x - root)
itp(trig1, c(-1, 1), root = 1 / 10)
# Supplying an external pointer to a C++ function
trig1_ptr <- xptr_create("trig1")
itp(f = trig1_ptr, c(-1, 1), root = 1 / 10)

# Logarithmic
logarithmic <- function(x, shift) log(abs(x - shift))
itp(logarithmic, c(-1, 1), shift = 10 /9)

# Linear
linear <- function(x) x
# Solution in one iteration
itp(linear, c(-1, 1))
# Solution at an input endpoint
itp(linear, c(-1, 0))

### Ill-behaved functions

## Non-simple zero
```

```
# Polynomial 3
poly3 <- function(x) (x * 1e6 - 1) ^ 3
itp(poly3, c(-1, 1))
# Using n0 = 0 leads to fewer iterations, in this example
poly3 <- function(x) (x * 1e6 - 1) ^ 3
itp(poly3, c(-1, 1), n0 = 0)

## Discontinuous

# Staircase
staircase <- function(x) ceiling(10 * x - 1) + 1 / 2
itp(staircase, c(-1, 1))

## Multiple roots

# Warsaw
warsaw <- function(x) ifelse(x > -1, sin(1 / (x + 1)), -1)
# Function increasing over the interval
itp(warsaw, c(-1, 1))
# Function decreasing over the interval
itp(warsaw, c(-0.85, -0.8))
```

`itp_c` *The ITP root-finding algorithm using C++*

Description

Performs one-dimensional root-finding using the ITP algorithm of Oliveira and Takahashi (2021). This function is equivalent to `itp` but calculations are performed entirely using C++, and the arguments differ slightly: `itp_c` has a named required argument `pars` rather than `...` and it does not have the arguments `interval`, `f.a` or `f.b`.

Usage

```
itp_c(f, pars, a, b, epsilon = 1e-10, k1 = -1, k2 = 2, n0 = 1)
```

Arguments

`f`	An external pointer to a C++ function that evaluates the function f.
`pars`	A list of additional arguments to the function. This may be an empty list.
`a, b`	Numeric scalars. Lower (`a`) and upper `b` limits of the interval to be searched for a root.
`epsilon`	A positive numeric scalar. The desired accuracy of the root. The algorithm continues until the width of the bracketing interval for the root is less than or equal to `2 * epsilon`.

k1, k2, n0 — Numeric scalars. The values of the tuning parameters κ_1, κ_2, n_0. See the **Details** section of itp.

The default value for k1 in itp_c is set as the inadmissible value of -1 (in reality κ_1 must be positive) as a device to set the same default value for k1 as itp, that is, k1 = 0.2 / (b - a). If the input value of k1 is less than or equal to 0 then, inside itp_c, k1 = 0.2 / (b - a) is set.

Details

For details see itp.

Value

An object (a list) of class "itp" with the same structure as detailed in the **Value** section of itp, except that the attribute f_name is empty (equal to "").

References

Oliveira, I. F. D. and Takahashi, R. H. C. (2021). An Enhancement of the Bisection Method Average Performance Preserving Minmax Optimality, *ACM Transactions on Mathematical Software*, **47**(1), 1-24. doi:10.1145/3423597

See Also

print.itp and plot.itp for print and plot methods for objects of class "itp" returned from itp_c or itp.

Examples

```
wiki_ptr <- xptr_create("wiki")
wres <- itp_c(f = wiki_ptr, pars = list(), a = 1, b = 2, epsilon = 0.0005)
wres
plot(wres, main = "Wiki")
```

---

plot.itp — *Plot method for objects of class "itp"*

Description

Plot method for objects of class "itp" returned from itp.

Usage

```
## S3 method for class 'itp'
plot(x, ...)
```

Arguments

x	An object inheriting from class "itp", a result of a call to itp.
...	Arguments passed to curve, such as graphical parameters.

Details

Uses curve to produce a plot of the function f provided to itp over the interval within which a root was sought. The estimated root is indicated using a horizontal line drawn at 0 and a vertical line drawn at the estimated root. By default the name of the function f is used as a title, but this can be replaced by supplying the argument main. The interval over which f is plotted can be changed by supplying the arguments from and/or to.

Value

No return value, only the plot is produced.

See Also

itp for the Interpolate, Truncate, Project (ITP) root finding algorithm.

Examples

```
# Lambert

# Supplying an R function
lambert <- function(x) x * exp(x) - 1
x <- itp(lambert, c(-1, 1))
plot(x)

# Supplying an external pointer to a C++ function
lambert_ptr <- xptr_create("lambert")
x <- itp(lambert_ptr, c(-1, 1))
plot(x, main = "Lambert")
```

---

print.itp *Print method for objects of class "itp"*

Description

Prints objects of class "itp" returned from itp.

Usage

```
## S3 method for class 'itp'
print(x, all = FALSE, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

x	An object inheriting from class "itp", a result of a call to itp.
all	A logical scalar. If all = FALSE then only the estimated root, the value of the function at the root an the number of iterations are printed. If all = TRUE then, in addition, the final bracketing interval [a, b], the values of the function at the end points of this interval and the estimated precision are printed.
digits	The argument digits passed to format to set the number of significant digits to print.
...	Further arguments to be passed to or from other methods. They are ignored in this function..

Details

The default setting is to print only the root, the value of the function at the root and the number of iterations. To include the bracketing interval after convergence and the estimated precision use all = TRUE.

Value

The argument x is returned, invisibly.

See Also

itp for the Interpolate, Truncate, Project (ITP) root finding algorithm.

---

xptr_create *Create an external pointer to a C++ function*

Description

This function is used in the itp package to create external pointers to the C++ functions used as examples to illustrate the use of the function itp. These pointers are passed as the argument f to itp. To create their own examples the user will need to create their own C++ function(s) and a function that is similar to xptr_create.

Usage

```
xptr_create(fstr)
```

Arguments

fstr	A string indicating the C++ function required.

Details

See the vignette Overview of the itp package and the file user_fns.cpp for information.

The example C++ functions available in itp are: "wiki", "lambert", "trig1", "poly3", "linear", "warsaw" and staircase.

Value

The external pointer.

See Also

xptr_eval for calling a C++ function using an external pointer.

Examples

```
lambert_ptr <- xptr_create("lambert")
res <- itp(lambert_ptr, c(-1, 1))
```

---

xptr_eval *Call a C++ function using an external pointer*

Description

This function is used in plot.itp to plot a function and the root estimated by itp.

Usage

```
xptr_eval(x, pars, xpsexp)
```

Arguments

x	The main argument of the function.
pars	A list of additional arguments to the function. This may be an empty list.
xpsexp	An external pointer to a C++ function.

Details

See the Passing user-supplied C++ functions article in the Rcpp Gallery for information.

Value

A numeric scalar: the value of the C++ function evaluated at the input values x and pars.

See Also

xptr_create for creating an external pointer to a C++ function.

Examples

```
lambert_ptr <- xptr_create("lambert")
res <- itp(lambert_ptr, c(-1, 1))

# Value at lower limit
xptr_eval(-1, list(), lambert_ptr)

# Value at upper limit
xptr_eval(1, list(), lambert_ptr)

# Value at the estimated root
xptr_eval(res$root, list(), lambert_ptr)
```

Index