Package 'ktsolve'

May 8, 2026

Type Package

Title Configurable Function for Solving Families of Nonlinear Equations

Version 1.4

Date 2026-01-30

Maintainer Carl Witthoft <cellocgw@gmail.com>

Description This is designed for use with an arbitrary set of equations with an arbitrary set of unknowns.
The user selects ``fixed'' values for enough unknowns to leave as many variables as there are equations, which in most cases means the system is properly defined and a unique solution exists. The function, the fixed values and initial values for the remaining unknowns are fed to a nonlinear backsolver.
The original version of ``TK!Solver'' , now a product of Universal Technical Systems (<https://www.uts.com>) was the inspiration for this function.

License LGPL-3

Imports methods, BB, nleqslv, rootSolve

NeedsCompilation no

Author Carl Witthoft [aut, cre]

Repository CRAN

Date/Publication 2026-01-31 14:10:02 UTC

Contents

ktsolve-package *Configurable Function for Solving Families of Nonlinear Equations*

Description

This function is designed for use with an arbitrary set of equations with an arbitrary set of unknowns. The user selects "fixed" values for enough unknowns to leave as many variables as there are equations, which in most cases means the system is properly defined and a unique solution exists. The function, the fixed values, and initial values for the remaining unknowns are fed to a nonlinear backsolver. The original version of "TK!Solver" was the inspiration for this function.

Details

Package:	ktsolve
Type:	Package
Version:	1.4
Date:	2026-01-30
License:	GPL-3

Note: `ktsolve` requires at least one of the "Suggests" packages (currently `BB` , `nleqslv` , and `rootSolve`) to execute nonlinear back-solvers.

Author(s)

Carl Witthoft

Maintainer: Carl Witthoft, carl@witthoft.com

ktsolve *Configurable Function for Solving Families of Nonlinear Equations*
Version: 1.4

Description

This function is designed for use with an arbitrary set of equations with an arbitrary set of unknowns. The user selects "fixed" values for enough unknowns to leave as many variables as there are equations, which in most cases means the system is properly defined and a unique solution exists. The function, the fixed values, and initial values for the remaining unknowns are fed to a nonlinear backsolver. As of version 1.3, supports `BB` and `nleqslv`

Usage

```
ktsolve(yfunc, known = list(), guess,
tool = c("BBsolve", "nleqslv","multiroot"), show = TRUE, ...)
```

Arguments

yfunc	a function which accepts a vector of length n and produces an output vector of length n. See the rules for constructing yfunc below.
known	A list of known values. The elements must be named and the names must match variable names in yfunc.
guess	A list or vector of initial guesses for the unknown values. The elements must be named and the names must match variable names in yfunc. AND length(guess) must be same as the number of y[j] equations in yfunc, to avoid having an over- or under-defined system.
tool	name of package which holds the solver to be used. Currently only BB::BBsolve, nleqslv::nleqslv, and rootSolve::multiroot are supported.
show	if TRUE, the solution values are printed in brief form to the console.
...	additional arguments to pass to the selected tool package.

Details

The input form of yfunc is a function of the form:

```
 yfunc<-function(x) {
y<-vector()
y[1]<-f1(known,guess)
y[2]<-f2(known,guess)
.
.
.
y[n]<-fn(known,guess)
}
```

where y[j] are dummies which will be driven to zero, and x is a dummy vector which is used (with the "guess" values) to run the solver. So, eqns in the form `A=f(x)` must be entered as `y[j] <- f(x)-A`

For example, `d = a + sqrt(b)` and `a = sin(a/b) + g*exp(f*a)` become

`y[1]<- a - d +sqrt(b)` and `y[2]<- sin(a/b) +g*exp(f*a) -a` ,

and e.g. `known <- list(a=3,d=5,g=.1)` are the fixed parameters and

`guess <- list(b=1,f=1)` are the initializers for the solver.

Note that it is not necessary to have any known values if the function in question has as many (independent) equations as unknowns. One of the handy things about ktsolve is the ease with which one can swap 'known' and 'guess' inputs to evaluate the system over different parameter sets.

Value

results	The output returned from the called solver package. As such, the contents and structure depend on which package was invoked via the tools argument.
tool	Echoes back the selected solver package used for reference.
yfunc	Returns the modified yfunc as a function for the user to review and /or use to process additional data, using the fit values generated.

Note

The original version of TK!Solver provided a very nice GUI-based version of what I've done in `ktsolver`. Over the years, it's turned into a very large, powerful, and,sadly, expensive application. You can find it at https://www.uts.com/Products/Tksolver

Author(s)

Carl Witthoft, <carl@witthoft.com>

See Also

BBsolve , nleqslv

Examples

```
zfunc<-function(x) {
z<-vector()
z[1]<- 4*var1 -3*var2 +5*var3
z[2]<-8*var1 +5*var2 -2*var3
z
}

 known=list(var2=5)
 guess=list(var1=2,var3=0)
 solv1 <- ktsolve(zfunc,known=known,guess=guess)
#   Successful convergence.
# solution is:
#      var1      var3
# -1.979167  4.583333
# "known" inputs were:
#       var2
# known 5
 eval(solv1$yfunc)(solv1$results$par)


 known=list(var1=5)
 guess=list(var2=2,var3=0)
 solv2<- ktsolve(zfunc,known=known,guess=guess)
#   Successful convergence.
# solution is:
#      var2      var3
# -12.63158 -11.57895
# "known" inputs were:
#       var1
# known 5
eval(solv2$yfunc)(solv2$results$par)
```

Index