Package 'gripp'

May 8, 2026

Type Package

Title General Inverse Problem Platform

Version 0.2.21

Maintainer Jader Lugon Junior <jlugonjr@gmail.com>

Description Set of functions designed to solve inverse problems. The direct problem is used to calculate a cost function to be minimized.
Here are listed some papers using Inverse Problems solvers and sensitivity analysis:
(Jader Lugon Jr.; Antonio J. Silva Neto 2011) <doi:10.1590/S1678-58782011000400003>.
(Jader Lugon Jr.; Antonio J. Silva Neto; Pedro P.G.W. Rodrigues 2008) <doi:10.1080/17415970802082864>.
(Jader Lugon Jr.; Antonio J. Silva Neto; Cesar C. Santana 2008) <doi:10.1080/17415970802082922>.

Depends R (>= 2.10)

Imports utils, GenSA, GA

License GPL-3

Encoding UTF-8

Repository CRAN

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.3.3

NeedsCompilation no

Author Jader Lugon Junior [aut, cre] (ORCID:
<https://orcid.org/0000-0001-8030-0713>),
Antonio Jose Silva Neto [aut] (ORCID:
<https://orcid.org/0000-0002-9616-6093>)

Date/Publication 2025-12-16 23:50:02 UTC

Contents

`cost` *Cost function*

Description

This cost function is the one to be minimized using the Inverse Problem solver. It will need some information about the target and the direct problem.

Usage

```
cost(parm)
```

Arguments

`parm`	Set of values to be considered as parameters for the Direct Problem solution

Value

Squared Difference between the target and the result that is calculated with the informed set of parameters 'parm'.

Examples

```
#
# Configure the name of the folder where each file with the parameter to be changed.
# If the folder is 'wd' then it will not be changed.
# You can change it to another location where your parameter files are.
auxi <- system.file(package = "gripp")
folder_name <- c(auxi,auxi)
#
# Configure the name of each file to be changed in the Direct Problem Solver.
file_name <- c('f1.R','f1.R')
#
# Configure the name of each parameter to be changed.
parm_name <- c('A','B')
#
# Configure alternative method using line number to enter each parameter
# When line_number is zero, then the input method uses keywords.
line_number <-c(0,0)
#
# Configure each parameter type, where:
#   1 means a numeric variable informed as a string, such as parm <- "1.38"
#  2 means a numeric vector informed as a string of numbers separated with a space character.
#    For example, parm <- "1.25 3.4". You must tell which one will be the variable considered.
parm_type <- c(1,1)
```

```
#
# Informe the position of the variable to be considered in the problem in the vector.
# If the parameter is a numeric variable, then its position is zero.
parm_vector <- c(0,0)
#
# Configure the name of the folder where the command must be called.
# If the folder is 'wd' then it will not be changed.
# You can change it to another location where your parameter files are.
command_folder <- auxi
#
# This variable means that the Direct Problem is solved using R or outside
# if this is FALSE, the results are to be read from a file.
# When it is TRUE, results will be passed inside R using the variable "results"
isitR <- TRUE
#
# Configure the command to be used to call the Direct Problem Solver.
command <- 'f1.R'
#
# Configure the name of the file to be used as a target by the cost function.
# The file was built using parm <- c(1,3).
# It is also a single column of values.
target <- 'alvo.dat'
#
# Configure the name of the folder where the target file can be found.
# If the folder is 'wd' then it will not be changed.
# You can change it to another location where your target file is.
target_folder <- auxi
#
# Configure a string to be used to attrib values in the Direct Problem file.
attrib_str <- '<-'
#
# Configure the name of the file with the results obtained by the Direct Problem.
# It must be a single column of values.
# The results are changed at each run by the Direct Problem Solver.
result <- 'result.dat'
#
# Configure the name of the folder where the results can be found after each run.
# If the folder is 'wd' then it will not be changed.
# You can change it to another location where your result file is.
result_folder <- tempdir()
#
parm<- c(1,3)
cost(parm)
parm<- c(1.5,4)
cost(parm)
```

---

invprob *Inverse Problem Solver!*

Description

This sotfware will minimize a cost function and estimate a set of parameters using a Inverse Problem Solver.

Usage

```
invprob(parm_init)
```

Arguments

parm_init	Set of values to be considered as initial parameters for the Inverse Problem solution

Value

A vector with the quadratic residue calculated, the parameters estimated and the number of function call needed to solve the inverse problem.

Examples

```
#
# Configure the name of the folder where each file with the parameter to be changed.
# If the folder is 'wd' then it will not be changed.
# You can change it to another location where your parameter files are.
auxi <- system.file(package = "gripp")
folder_name <- c(auxi,auxi)
#
# Configure the name of each file to be changed in the Direct Problem Solver.
file_name <- c('f1.R','f1.R')
#
# Configure the name of each parameter to be changed.
parm_name <- c('A','B')
#
# Configure alternative method using line number to enter each parameter
# When line_number is zero, then the input method uses keywords.
line_number <-c(0,0)
#
# Configure each parameter type, where:
#   1 means a numeric variable informed as a string, such as parm <- "1.38"
#  2 means a numeric vector informed as a string of numbers separated with a space character.
#   For example, parm <- "1.25 3.4". You must tell which one will be the variable considered.
parm_type <- c(1,1)
#
# Informe the position of the variable to be considered in the problem in the vector.
# If the parameter is a numeric variable, then its position is zero.
parm_vector <- c(0,0)
#
# Configure the smallest value for each parameter
parm_min <- c(0,0)
#
# Configure the larger value for each parameter
```

```
parm_max <- c(2,5)
#
# Configure the name of the folder where the command must be called.
# If the folder is 'wd' then it will not be changed.
# You can change it to another location where your parameter files are.
command_folder <- system.file(package = "gripp")
#
# This variable means that the Direct Problem is solved using R or outside
# if this is FALSE, the results are to be read from a file.
# When it is TRUE, results will be passed inside R using the variable "result"
isitR <- TRUE
#
# Configure the command to be used to call the Direct Problem Solver.
command <- 'f1.R'
#
# Configure a string to be used to attrib values in the Direct Problem file.
attrib_str <- '<-'
#
# Configure the name of the file with the results obtained by the Direct Problem.
# It must be a single column of values.
# The results are changed at each run by the Direct Problem Solver.
result <- 'result.dat'
#
# Configure the name of the folder where the results can be found after each run.
# If the folder is 'wd' then it will not be changed.
# You can change it to another location where your result file is.
result_folder <- tempdir()
#
# Configure the name of the file to be used as a target by the cost function.
# The file was built using parm <- c(1,3).
# It is also a single column of values.
target <- 'alvo.dat'
#
# Configure the name of the folder where the target file can be found.
# If the folder is 'wd' then it will not be changed.
# You can change it to another location where your target file is.
target_folder <- auxi
#
# Configure the Inverse Problem Solver to be used:
# 1) GRIPP can use the solver 'GenSA'.
# Then configure the maximum number of function evaluation for the Inverse Problem Solution
# using control <- list(max.time=60)
#
# 2) GRIPP can use the solver 'GA'
# The configure the maximum number of function evaluation for the Inverse Problem Solution
# control <- list(popSize = 50, maxiter = 1000, run = 100)
solver <- 'GenSA'
control <- list(max.time=1)
# Configure the initial value for each parameter
parm <- c(1.5,4)
out<-invprob(parm)
out
```

sensitivity *Sensitivity matrix calculator*

Description

This sotfware will calculate the sensitivity matrix for the Direct Problem. First order derivatives are calculated using central difference approximation.

Usage

```
sensitivity(parm_s)
```

Arguments

parm_s	Set of values to be considered as parameters for the Direct Problem solution

Value

A matrix with the derivative of the function that represents the Direct Problem for each parameter.

Examples

```
#
# Configure the name of the folder where each file with the parameter to be changed.
# If the folder is 'wd' then it will not be changed.
# You can change it to another location where your parameter files are.
auxi <- system.file(package = "gripp")
folder_name <- c(auxi,auxi)
#
# Configure the name of each file to be changed in the Direct Problem Solver.
file_name <- c('f1.R','f1.R')
#
# Configure the name of each parameter to be changed.
parm_name <- c('A','B')
#
# Configure alternative method using line number to enter each parameter
# When line_number is zero, then the input method uses keywords.
line_number <-c(0,0)
#
# Configure each parameter type, where:
#   1 means a numeric variable informed as a string, such as parm <- "1.38"
#  2 means a numeric vector informed as a string of numbers separated with a space character.
#   For example, parm <- "1.25 3.4". You must tell which one will be the variable considered.
parm_type <- c(1,1)
#
# Informe the position of the variable to be considered in the problem in the vector.
# If the parameter is a numeric variable, then its position is zero.
parm_vector <- c(0,0)
#
# Configure the smallest value for each parameter
```

```
parm_min <- c(0,0)
#
# Configure the larger value for each parameter
parm_max <- c(2,5)
#
# Configure the name of the folder where the command must be called.
# If the folder is 'wd' then it will not be changed.
# You can change it to another location where your parameter files are.
command_folder <- auxi
#
# This variable means that the Direct Problem is solved using R or outside
# if this is FALSE, the results are to be read from a file.
# When it is TRUE, results will be passed inside R using the variable "results"
isitR <- TRUE
#
# Configure the command to be used to call the Direct Problem Solver.
# if this is FALSE, the results are to be read from a file.
# When it is TRUE, results will be passed inside R using the variable "result"
command <- 'f1.R'
#
# Parameter positive and negative percentual difference to be used to calculate the derivative
# ppdif must me a number between 0 and 100
# parameter_pos <- parm + (ppdif/100)*(parm_max-parm_min)
# parameter_neg <- parm - (ppdif/100)*(parm_max-parm_min)
ppdif <- 1
#
# Configure a string to be used to attrib values in the Direct Problem file.
attrib_str <- '<-'
#
# Configure the name of the file with the results obtained by the Direct Problem.
# It must be a single column of values.
# The results are changed at each run by the Direct Problem Solver.
result <- 'result.dat'
#
# Configure the name of the folder where the results can be found after each run.
# If the folder is 'wd' then it will not be changed.
# You can change it to another location where your result file is.
result_folder <- tempdir()
#
sensitivity(c(1,3))
```

`synthetic` *Synthetic Experimental Data function*

Description

The Direct Problem will be solved and the result will be corrupted to simulate experimental error. The user can inform the Standard Deviation error that will be added to the exact solution. Then, the synthetic function will return the data obtained using the equation:

Simulated_Results_with_Error <- Direct_Problem_Results * (1 + sigma).

Usage

```
synthetic(parm,sigma)
```

Arguments

parm	Values for each parameter needed to solve the Direct Problem.
sigma	Standard Deviation of the synthetic data produced using the Direct Problem solution.

Value

The result will be saved in the result file.

Examples

```
#
# Configure the name of the folder where each file with the parameter to be changed.
# If the folder is 'wd' then it will not be changed.
# You can change it to another location where your parameter files are.
auxi <- system.file(package = "gripp")
folder_name <- c(auxi,auxi)
#
# Configure the name of each file to be changed in the Direct Problem Solver.
file_name <- c('f1.R','f1.R')
#
# Configure the name of each parameter to be changed.
parm_name <- c('A','B')
#
# Configure alternative method using line number to enter each parameter
# When line_number is zero, then the input method uses keywords.
line_number <-c(0,0)
#
# Configure each parameter type, where:
#   1 means a numeric variable informed as a string, such as parm <- "1.38"
#  2 means a numeric vector informed as a string of numbers separated with a space character.
#   For example, parm <- "1.25 3.4". You must tell which one will be the variable considered.
parm_type <- c(1,1)
#
# Informe the position of the variable to be considered in the problem in the vector.
# If the parameter is a numeric variable, then its position is zero.
parm_vector <- c(0,0)
#
# Configure the name of the folder where the command must be called.
# If the folder is 'wd' then it will not be changed.
# You can change it to another location where your parameter files are.
command_folder <- auxi
#
# This variable means that the Direct Problem is solved using R or outside
# if this is FALSE, the results are to be read from a file.
# When it is TRUE, results will be passed inside R using the variable "results"
isitR <- TRUE
```

```
#
# Configure the command to be used to call the Direct Problem Solver.
command <- 'f1.R'
#
# Configure a string to be used to attrib values in the Direct Problem file.
attrib_str <- '<-'
#
# Configure the name of the file with the results obtained by the Direct Problem.
# It must be a single column of values.
# The results are changed at each run by the Direct Problem Solver.
result <- 'result.dat'
#
# Configure the name of the folder where the results can be found after each run.
# If the folder is 'wd' then it will not be changed.
# You can change it to another location where your result file is.
result_folder <- tempdir()
#
parm<- rep(5,3)
synthetic(parm,0.03)
```

Index