Package 'FixedPoint'

June 5, 2018

Type Package

Title Algorithms for Finding Fixed Point Vectors of Functions

Version 0.5

Author Stuart Baumann & Margaryta Klymak

Maintainer Stuart Baumann <Stuart@StuartBaumann.com>

Description
For functions that take and return vectors (or scalars), this package provides 8 algorithms for finding fixed point vectors (vectors for which the inputs and outputs to the function are the same vector). These algorithms include Anderson (1965) acceleration <doi:10.1145/321296.321305>, epsilon extrapolation methods (Wynn 1962 <doi:10.2307/2004051>) and minimal polynomial methods (Cabay and Jackson 1976 <doi:10.1137/0713060>).

License MIT + file LICENSE

Depends MASS

Suggests testthat, schumaker, cubature, knitr, foreach, doParallel, SQUAREM

LazyData TRUE

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-06-05 21:33:58 UTC

R topics documented:

ChangePerIterate 2
ConvergenceFig 3
EpsilonExtrapolation 4
EpsilonExtrapolationVectorOfInverses 5
FixedPoint 5
FixedPointNewInput 8
PolynomialExtrapolation 9
PutTogetherIteratesWithoutJumps 10

Index 12

`ChangePerIterate` *ChangePerIterate A function for plotting the change in each vector per iterate.*

Description

ChangePerIterate A function for plotting the change in each vector per iterate.

Usage

```
ChangePerIterate(Inputs, Outputs, ConvergenceVector = c(),
  ConvergenceSigFig = 5, ShowInputs = TRUE, ShowOutputs = TRUE,
  ShowPrevious = TRUE, xaxis = c(), secondhold = 0, FromIterate = 1,
  ToIterate = c())
```

Arguments

`Inputs`	The Inputs matrix returned by FixedPoint.
`Outputs`	The Outputs matrix returned by FixedPoint.
`ConvergenceVector`	The Convergence vector returned by fixedpoint.
`ConvergenceSigFig`	The number of significant figures convergence values should be shown with in the plot header.
`ShowInputs`	A boolean describing whether to show the inputs in the figure.
`ShowOutputs`	A boolean describing whether to show the outputs in the figure.
`ShowPrevious`	A boolean describing whether to show the previous inputs/outputs in the figure.
`xaxis`	A vector for meaningful values corresponding to the input/output values. By default the indexes of each input/output vector are used.
`secondhold`	If this is -1 or less then all plotting happens immediately. This means that very quickly only the last iterate will be seen so it makes sense to do it only if FromIterate and ToIterate indicate only one iterate. If this is 0 then a user can click through each figure. If this is positive then it describes how many seconds to pause between frames. By default this is 0 so a user must click through each frame.
`FromIterate`	This describes what iterate to show first.
`ToIterate`	This describes what iterate to show last.

Value

This function returns nothing. It just shows a plot in the console.

Examples

```
Inputs = seq(1,10)
Function = function(x){ cos(x) }
A = FixedPoint(Function, Inputs, Method = "Anderson")
ChangePerIterate(A$Inputs, A$Outputs, A$Convergence)
#Any now to have it play one frame every half a second starting from the nineth iterate
ChangePerIterate(A$Inputs, A$Outputs, A$Convergence, secondhold = 0.5, FromIterate = 9)
```

ConvergenceFig *A function for plotting the convergence change over a series of iterates.*

Description

On the x axis is the index of each iterate. On the y axis are different convergence metric values.

Usage

```
ConvergenceFig(Inputs, Outputs, Input_Convergence = c(),
  LinesToDraw = c("Sup_Norm", "Euclidean_Norm",
  "Sum_Of_Absolute_Value_Of_Residuals", "Smallest_Residual"), LogScale = TRUE,
  FromIterate = 1, ToIterate = c())
```

Arguments

Inputs	The Inputs matrix returned by FixedPoint.
Outputs	The Outputs matrix returned by FixedPoint.
Input_Convergence	Convergence vector to be used in the figure. Often makes sense for it to be the same as that used in the Fixedpoint function.
LinesToDraw	A vector describing which convergence metrics to draw. This vector can contain anything in c("Sup_Norm", "Euclidean_Norm", "Sum_Of_Absolute_Value_Of_Residuals", "Smallest_Residual", "Input_Convergence"). Here "Input_Convergence" draws the input convergence vector. "Sup_Norm" gives the greatest residual (in absolute value), "Euclidean_Norm" is the eucleadean norm of the residuals, "Sum_Of_Absolute_Value_Of_Re sums up the absolute value of the residuals and "Smallest_Residual" is the smallest residual by absolute value. Clearly if the function takes and returns a scalar (rather than a vector) then all of these are identical.
LogScale	This is whether or not to use a log scale. If so instead of convergence, log(1+convergence) is shown on the y axis. By default this is true.
FromIterate	This describes the first iterate on the figure.
ToIterate	This describes the last iterate on the figure.

Value

This function returns nothing. It just shows a plot in the console.

Examples

```
Inputs = seq(1,10)
Function = function(x){ cos(x) }
A = FixedPoint(Function, Inputs, Method = "Anderson")
ConvergenceFig(A$Inputs, A$Outputs)
# And now to only show a median norm:
Differences = A$Inputs - A$Outputs
Convergence = apply(Differences,2,function(x){median(abs(x))})
ConvergenceFig(A$Inputs, A$Outputs, Convergence, LinesToDraw = "Input_Convergence")
```

`EpsilonExtrapolation` *EpsilonExtrapolation This function takes a matrix with previous iterates and extrapolates the limit of the sequence.*

Description

EpsilonExtrapolation This function takes a matrix with previous iterates and extrapolates the limit of the sequence.

Usage

```
EpsilonExtrapolation(Iterates, Method = c("VEA", "SEA"))
```

Arguments

`Iterates`	A matrix representing different iterates with one iterate per column. Can be pieced together from Inputs and Outputs matrices of the FixedPoint function using the PutTogetherIteratesWithoutJumps function.
`Method`	Method for epsilon extrapolation. Should be either "VEA" for the vector extrapolation algorithm or "SEA" for the scalar epsilon algorithm.

Value

A vector with the extrapolated vector.

Examples

```
FPFunction = function(x){c(0.5*sqrt(abs(x[1] + x[2])), 1.5*x[1] + 0.5*x[2])}
A = FixedPoint( Function = FPFunction, Inputs = c(0.3,900), MaxIter = 6, Method = "Simple")
Iterates = PutTogetherIteratesWithoutJumps(A$Inputs, A$Outputs)
EpsilonExtrapolation(Iterates, "VEA")
B = FixedPoint( function(x){cos(x)}, Inputs = 1, MaxIter = 5, Method = "Simple")
Iterates = PutTogetherIteratesWithoutJumps(B$Inputs, B$Outputs)
EpsilonExtrapolation(Iterates, "SEA")
```

EpsilonExtrapolationVectorOfInverses

EpsilonExtrapolationVectorOfInverses This is a helper function for EpsilonExtrapolation

Description

EpsilonExtrapolationVectorOfInverses This is a helper function for EpsilonExtrapolation

Usage

```
EpsilonExtrapolationVectorOfInverses(DifferenceMatrix, Method)
```

Arguments

DifferenceMatrix	The matrix of the differences in elements to be inverted.
Method	"SEA" or "VEA".

Value

A vector of the result of inverting each (column) vector in a mmatrix.

FixedPoint

A function for finding the fixed point of a contraction mapping

Description

This function takes in a function and an initial guess for the fixed point of that function. It then uses a fixed point method to find the fixed point of that function.

Usage

```
FixedPoint(Function, Inputs, Outputs = c(), Method = c("Anderson", "Simple",
  "Aitken", "Newton", "MPE", "RRE", "VEA", "SEA"),
  ConvergenceMetric = function(Residuals) {     max(abs(Residuals)) },
  ConvergenceMetricThreshold = 1e-10, MaxIter = 1000, MaxM = 10,
  ExtrapolationPeriod = 7, Dampening = 1,
  ReplaceInvalids = c("ReplaceElements", "ReplaceVector", "NoAction"),
  PrintReports = FALSE, ReportingSigFig = 5,
  ConditionNumberThreshold = 1000, Plot = c("NoPlot", "ConvergenceFig",
  "ChangePerIterate"), ConvergenceFigLags = 5, ChangePerIteratexaxis = c())
```

Arguments

`Function`	This is the function for which a fixed point is sought. This function must take and return a vector of the same dimension.
`Inputs`	This can be either a vector of values that is an initial guess for a fixed point or it can be an N x A matrix of previous inputs for which corresponding outputs are available. In this case N is the dimensionality of the fixed point vector you are seeking (Hence each column is a matrix that is input to the "Function") and A is the number of previous Inputs/Outputs that are being provided to the fixed point. Where a matrix is input, a corresponding outputs must be provided or the last column of the outputs matrix is taken as a startpoint guess and the rest of the inputs and output matrices are discarded.
`Outputs`	(Optional) This is a matrix of the Function values for each column of the input. It must be provided so that column k of the outputs matrix is equal to Function(Column k of inputs matrix).
`Method`	This is the fixed point method to be used. It can be "Anderson", "Simple", "Aitken", "Newton", "MPE", "RRE", "VEA" or "SEA". See vignette and references to see explanations of these methods.
`ConvergenceMetric`	This is a function that takes in a vector of residuals from one iterate of the function (defined as f(x) - x for vector x and function f) and returns a scalar. This scalar should be low when convergence is close to being achieved. By default this is the maximum residual by absolute value (the sup norm in the space of residuals).
`ConvergenceMetricThreshold`	This is the threshold for terminating the algorithm. The algorithm will terminate when the scalar that ConvergenceMetric returns is less than ConvergenceMetricThreshold. This can be set to a negative number in which case the algorithm will run until MaxIter is hit or an error occurs (Note that an error is likely in trying to use any method other than "Simple" when a fixed point is already found).
`MaxIter`	This is the maximum number of iterates that will be undertaken.
`MaxM`	This is the maximum number of saved iterates that are used in the Anderson algorithm. It has no effect if another method is chosen. Note that the number of previous iterates that will actually be used is the minimum of MaxIter, the dimensionality of the function's vector and the number of inputs that have been tried to previously (the width of the Outputs matrix at each given stage of the algorithm). If PrintReports = TRUE, the number of previous iterates actually used is reported as the algorithm is running.
`ExtrapolationPeriod`	This is the number of simple iterates to perform before extrapolating. This is used for the MPE, RRE, VEA and SEA methods and has no effect if another method is chosen. Where an epsilon algorithm is used this should be one plus a multiple of two, ie (3,5,7,etc).
`Dampening`	This is the dampening parameter. By default it is 1 which means no dampening takes place. It can also be less than 1 (indicating dampening) or more than 1 (indicating extrapolation).

`ReplaceInvalids`
: This determines how NAs and Infs generated by the extrapolation method are handled. If it is "NoAction" then if the extrapolation algorithm generates an NA or Inf then no action is taken. This will likely result in these NAs and Infs being fed into the function and an error resulting. If it is "ReplaceVector" then the entire extrapolated vector will be replaced by a vector of a simple iterate. If it is "ReplaceElements" then the elements containing an NA or Inf will be replaced by the corresponding elements from a simple iterate.

`PrintReports`
: This is a boolean describing whether to print ongoing ConvergenceMetric values for each iterate.

`ReportingSigFig`
: This is the number of significant figures that will be used in printing the convergence values to the console (only if PrintReports is TRUE).

`ConditionNumberThreshold`
: This is a threshold for what condition number is acceptable for solving the least squares problem for the Anderson Method. If the condition number is larger than this threshold then fewer previous iterates will be used in solving the problem. This has no effect unless the "Anderson" method is used.

`Plot`
: This determines whether a plot should be drawn for every iterate. It can be "NoPlot", "ConvergenceFig" or "ChangePerIterate". By default it is "NoPlot" and no plot is drawn. If it is "ConvergenceFig" then a plot is shown with iterates on the x axis and convergence (as defined by the ConvergenceMetric) is on the y axis. If it is "ChangePerIterate" then there is the index of the array value on the x axis and the value of the array value on the y axis. The previous iterate is also shown so the change per iterate can be visualised.

`ConvergenceFigLags`
: This only affects anything if Plot == "ConvergenceFig". This gives how many previous iterates should be shown on the x axis. By default it is 5. To see them all set it to a high number.

`ChangePerIteratexaxis`
: This only affects anything if Plot == "ChangePerIterate". Sometimes there is a more appropriate xaxis value to use than (the default) value index for this figure. For instance in the consumption smoothing problem in the vignette every value is a value function value at a given budget level. In this case the budget levels could be used for this xaxis.

Value

A list containing the FixedPoint, the Inputs and corresponding Outputs, and convergence values (which are computed under the "ConvergenceMetric"). The list will also include a "Finish" statement describing why it has finished. This is often going to be due to either MaxIter or ConvergenceMetricThreshold being reached. It may also terminate due to an error in generating a new input guess or using the function with that guess. If this occurs the function will terminate early and the "Finish" statement will describe the issue. In this event there will also be additional objects returned in the list "NewInputVector" and possibly "NewOutputVector" for use in debugging the issue.

Examples

```
# For the simplest possible example we can seek the fixed point of the cos function with a scalar.
Inputs = 0.3
Function = function(x){ cos(x) }
A = FixedPoint(Function, Inputs, Method = "Aitken", Dampening = 0.5)
B = FixedPoint(Function, Inputs, Method = "Anderson", Dampening = 1.0)

# For this next one the ConvergenceMetricThreshold is negative so the algorithm
# will keep running until MaxIter is met.
C = FixedPoint(Function, Inputs, Method = "Simple", MaxIter = 4, ConvergenceMetricThreshold = -1)
# This is not yet close to the fixed point but we can continue solving for this fixed point
# by inputting the previous inputs and outputs to the function. We can also switch methods
# and will switch below to the Newton Method.
D = FixedPoint(Function, C$Inputs, C$Outputs, Method = "Newton")

# We can also find a 4 dimensional fixed point vector of this function.
Inputs = c(0.3, 98, 0, pi)
E = FixedPoint(Function, Inputs, Method = "Anderson")
F = FixedPoint(Function, Inputs, Method = "Anderson", MaxM = 4, ReportingSigFig = 13)
```

`FixedPointNewInput` *FixedPointNewInput This function takes the previous inputs and outputs from the FixedPoint function and determines what vector to try next in seeking a fixed point.*

Description

FixedPointNewInput This function takes the previous inputs and outputs from the FixedPoint function and determines what vector to try next in seeking a fixed point.

Usage

```
FixedPointNewInput(Inputs, Outputs, Method = "Anderson", MaxM = 10,
  SimpleStartIndex = 1, ExtrapolationPeriod = 7, Dampening = 1,
  ConditionNumberThreshold = 1000, PrintReports = FALSE,
  ReplaceInvalids = c("ReplaceElements", "ReplaceVector", "NoAction"))
```

Arguments

`Inputs`	This is an N x A matrix of previous inputs for which corresponding outputs are available. In this case N is the dimensionality of the fixed point vector that is being sought (Hence each column is a matrix that is input to the "Function") and A is the number of previous Inputs/Outputs that are being provided to the fixed point.
`Outputs`	This is a matrix of Function values for the each column of the "Inputs" matrix.
`Method`	This is the fixed point method to be used. It can be "Anderson", "Simple", "Aitken", "Newton", "MPE", "RRE", "VEA", "SEA".

`MaxM` This is the number of saved iterates that are used in the Anderson algorithm. This has no role if another method is used.

`SimpleStartIndex`
This is the index for what column in the input/output matrices did the algorithm start doing simple iterates without jumps. This is used for all methods except the simple and Anderson methods where it has no effect.

`ExtrapolationPeriod`
This is the number of simple iterates to perform before extrapolating. This is used for the MPE, RRE, VEA and SEA methods and has no effect if another method is chosen. Where an epsilon algorithm is used this should be one plus a multiple of two, ie (3,5,7,etc).

`Dampening` This is the dampening parameter. By default it is 1 which means no dampening takes place. It can also be less than 1 (indicating dampening) or more than 1 (indicating extrapolation).

`ConditionNumberThreshold`
This is what threshold should be chosen to drop previous iterates if the matrix is ill conditioned. Only used in Anderson acceleration.

`PrintReports` This is a boolean describing whether to print ongoing ConvergenceMetric values for each iterate.

`ReplaceInvalids`
This determines how NAs and Infs generated by the extrapolation method are handled. If it is "NoAction" then if the extrapolation algorithm generates an NA or Inf then no action is taken. This will likely result in these NAs and Infs being fed into the function and an error resulting. If it is "ReplaceVector" then the entire extrapolated vector will be replaced by a vector of a simple iterate. If it is "ReplaceElements" then the elements containing an NA or Inf will be replaced by the corresponding elements from a simple iterate.

Value

A vector containing the next guess in seeking a fixed point.

Examples

```
FPFunction = function(x){c(0.5*sqrt(abs(x[1] + x[2])), 1.5*x[1] + 0.5*x[2])}
A = FixedPoint( Function = FPFunction, Inputs = c(0.3,900), MaxIter = 6, Method = "Simple")
NewGuessAnderson = FixedPointNewInput(A$Inputs, A$Outputs, Method = "Anderson")
NewGuessVEA = FixedPointNewInput(A$Inputs, A$Outputs, Method = "VEA")
NewGuessMPE = FixedPointNewInput(A$Inputs, A$Outputs, Method = "MPE")
NewGuessAitken = FixedPointNewInput(A$Inputs, A$Outputs, Method = "Aitken")
```

`PolynomialExtrapolation`
PolynomialExtrapolation This function performs Minimal Polynomial extrapolation (MPE) or Reduced Rank Extrapolation (RRE) given a matrix of previous iterates of the function.

Description

PolynomialExtrapolation This function performs Minimal Polynomial extrapolation (MPE) or Reduced Rank Extrapolation (RRE) given a matrix of previous iterates of the function.

Usage

```
PolynomialExtrapolation(Iterates, Method = c("MPE", "RRE"))
```

Arguments

Iterates	A matrix of inputs and outputs excluding jumps. Can be pieced together from Inputs and Outputs matrices of the FixedPoint function using the PutTogetherIteratesWithoutJumps function.
Method	The method for polynomial extrapolation. Should be either "MPE" for minimal polynomial extrapolation or "RRE" for reduced rank extrapolation.

Value

A vector containing the extrapolated vector.

Examples

```
FPFunction = function(x){c(0.5*sqrt(abs(x[1] + x[2])), 1.5*x[1] + 0.5*x[2])}
A = FixedPoint( Function = FPFunction, Inputs = c(0.3,900), MaxIter = 6, Method = "Simple")
Iterates = PutTogetherIteratesWithoutJumps(A$Inputs, A$Outputs)
PolynomialExtrapolation(Iterates, "MPE")
B = FixedPoint( function(x){cos(x)}, Inputs = 1, MaxIter = 5, Method = "Simple")
Iterates = PutTogetherIteratesWithoutJumps(B$Inputs, B$Outputs)
PolynomialExtrapolation(Iterates, "RRE")
```

PutTogetherIteratesWithoutJumps

PutTogetherIteratesWithoutJumps This function takes the previous inputs and outputs and assembles a matrix with both excluding jumps.

Description

PutTogetherIteratesWithoutJumps This function takes the previous inputs and outputs and assembles a matrix with both excluding jumps.

Usage

```
PutTogetherIteratesWithoutJumps(Inputs, Outputs, AgreementThreshold = 10 *
  .Machine$double.eps)
```

Arguments

Inputs	This is an N x A matrix of previous inputs for which corresponding outputs are available. In this case N is the dimensionality of the fixed point vector that is being sought (and each column is a matrix that is input to the "Function") and A is the number of previous Inputs/Outputs that are being provided to the fixed point.
Outputs	This is a matrix of "Function" values for each column of the "Inputs" matrix.
AgreementThreshold	A parameter for determining when a column in Inputs and a column in Outputs match. They are deemed to match if the sum of the absolute values of the difference in the columns is less than AgreementThreshold.

Value

A matrix of inputs and outputs excluding jumps.

Examples

```
A = FixedPoint( function(x){c(0.5*sqrt(abs(x[1] + x[2])), 1.5*x[1] + 0.5*x[2])},
Inputs = c(0.3,900), MaxIter = 5, Method = "Simple")
A = FixedPoint( function(x){c(0.5*sqrt(abs(x[1] + x[2])), 1.5*x[1] + 0.5*x[2])},
Inputs = A$Inputs, Outputs = A$Outputs, MaxIter = 5, Method = "Aitken")
A = FixedPoint( function(x){c(0.5*sqrt(abs(x[1] + x[2])), 1.5*x[1] + 0.5*x[2])},
Inputs = A$Inputs, Outputs = A$Outputs, MaxIter = 5, Method = "Simple")
CombinedSequence = PutTogetherIteratesWithoutJumps(A$Inputs, A$Outputs)
```

Index

ChangePerIterate, 2
ConvergenceFig, 3

EpsilonExtrapolation, 4
EpsilonExtrapolationVectorOfInverses, 5

FixedPoint, 5
FixedPointNewInput, 8

PolynomialExtrapolation, 9
PutTogetherIteratesWithoutJumps, 10