

# Package ‘AdvDif4’

May 6, 2026

**Type** Package

**Title** Solving 1D Advection Bi-Flux Diffusion Equation

**Version** 0.7.18

**Author**

Jader Lugon Junior, Pedro Paulo Gomes Watts Rodrigues, Luiz Bevilacqua, Gisele Moraes Marinho, Diego Campos Knupp, Joao Flavio Vieira Vasconcellos and Antonio Jose da Silva Neto.

**Maintainer** Jader Lugon Junior <jlugonjr@gmail.com>

**Description** This software solves an Advection Bi-Flux Diffusive Problem using the Finite Difference Method FDM. Vasconcellos, J.F.V., Marinho, G.M., Zanni, J.H., 2016, Numerical analysis of an anomalous diffusion with a bimodal flux distribution. <doi:10.1016/j.rimni.2016.05.001>. Silva, L.G., Knupp, D.C., Bevilacqua, L., Galeao, A.C.N.R., Silva Neto, A.J., 2014, Formulation and solution of an Inverse Anomalous Diffusion Problem with Stochastic Techniques. <doi:10.5902/2179460X13184>. In this version, it is possible to include a source as a function depending on space and time, that is,  $s(x,t)$ .

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**Repository** CRAN

**NeedsCompilation** no

**Date/Publication** 2019-07-21 18:20:02 UTC

## Contents

AdvDif4 . . . . .	2
pentaSolve . . . . .	7

<b>Index</b>	<b>9</b>
--------------	----------

**Description**

This software solves an Advection Bi-Flux Diffusive Problem using the Finite Difference Method FDM. A file with R commands can be consulted in document folder.

**Usage**

```
AdvDif4(parm, func)
```

**Arguments**

parm	Parameters data. It must contain values for k2,k4,v,l,m,tf,n,w10,w11,w12,w20,w21,w22,e10,e11,e12,e20 needed to run the model.
func	Functions definitions. It must contain the functions beta,dbetadp,fn,fs,fw1,fw2,fe1,fe2 needed to run the model.

**Value**

The resulting matrix with results obtained for each time as rows and at each position as columns.

**Examples**

```
#
# Begin of the first example
#
# 100th power sinusoidal function as initial condition and no source.
# with advection, bi-blux (primary and secondary diffusion) and constant beta.
#
# Beta function
fbeta <- function(p)
{f <- 0.2
return(f)}
# Beta derivative function
dbetadp <- function(p)
{f <- 0
return(f)}

# Initial condition
fn <- function(x)
{ f <- sin(pi*x)^100
return(f)}

# velocity
v <- 0.2

# Source function
```

```

fs <- function(x,t)
{ f <- 0
  return(f)}

# diffusion coefficients parameter
k2 <- 1e-3
k4 <- 1e-5

# Space and temporal definition
l <- 1
m <- 100
tf <- 1
n <- 1000

# Left boundary conditions
w10 <- 1
w11 <- 0
w12 <- 0
w20 <- 0
w21 <- 1
w22 <- 0
fw1 <- function(t)
{ f <- 0
  return(f)}
fw2 <- function(t)
{ f <- 0
  return(f)}

# Right boundary conditions
e10 <- 1
e11 <- 0
e12 <- 0
e20 <- 0
e21 <- 1
e22 <- 0
fe1 <- function(t)
{ f <- 0
  return(f)}
fe2 <- function(t)
{ f <- 0
  return(f)}

#
parm <- c(k2,k4,v,l,m,tf,n,w10,w11,w12,w20,w21,w22,e10,e11,e12,e20,e21,e22)
func <- c(fbeta=fbeta,dbetadp=dbetadp,fn=fn,fs=fs,fw1=fw1,fw2=fw2,fe1=fe1,fe2=fe2)
#
ad <- AdvDif4(parm,func)
eixo <- seq(0,1,by=0.01)
plot(eixo,ad[1,1:101],type='l',col="red",xaxt="n",xlab="X", ylab="p(x,t)")
axis(1,seq(0,1,0.1),las=2)
lines(eixo,ad[250,1:101],type='l',col="orange")
lines(eixo,ad[500,1:101],type='l',col="green")
lines(eixo,ad[750,1:101],type='l',col="blue")
lines(eixo,ad[1000,1:101],type='l',col="black")

```

```

#
#
# End of the first example
#
#
# Begin of the second example
# 100th power sinusoidal function as initial condition and no source.
# with advection, bi-blux (primary and secondary diffusion) and sigmoid function beta.
#
# Beta function
fbeta <- function(p)
{betamin <- 0.2
betamax <- 1
gama <- 2500
pin <- 0.001
f <- betamax-(betamax-betamin)/(1+exp(-gama*(p-pin)))
return(f)}
# Beta derivative function
dbetadp <- function(p)
{betamin <- 0.2
betamax <- 1
gama <- 2500
pin <- 0.001
f <- (-gama*(betamax-betamin)*exp(-gama*(p-pin))/((1+exp(-gama*(p-pin)))^2))
return(f)}

# Initial condition
fn <- function(x)
{ f <- sin(pi*x)^100
return(f)}

# velocity
v <- 0.2

# Source function
fs <- function(x,t)
{ f <- 0
return(f)}

# diffusion coefficients parameter
k2 <- 1e-3
k4 <- 1e-5

# Space and temporal definition
l <- 1
m <- 100
tf <- 1
n <- 1000

# Left boundary conditions
w10 <- 1
w11 <- 0
w12 <- 0

```

```

w20 <- 0
w21 <- 1
w22 <- 0
fw1 <- function(t)
{ f <- 0
  return(f)}
fw2 <- function(t)
{ f <- 0
  return(f)}

# Right boundary conditions
e10 <- 1
e11 <- 0
e12 <- 0
e20 <- 0
e21 <- 1
e22 <- 0
fe1 <- function(t)
{ f <- 0
  return(f)}
fe2 <- function(t)
{ f <- 0
  return(f)}
#
parm <- c(k2,k4,v,l,m,tf,n,w10,w11,w12,w20,w21,w22,e10,e11,e12,e20,e21,e22)
func <- c(fbeta=fbeta,dbetadp=dbetadp,fn=fn,fs=fs,fw1=fw1,fw2=fw2,fe1=fe1,fe2=fe2)
#
ad <- AdvDif4(parm,func)
eixo <- seq(0,1,by=0.01)
plot(eixo,ad[1,1:101],type='l',col="red",xaxt="n",xlab="X", ylab="p(x,t)")
axis(1,seq(0,1,0.1),las=2)
lines(eixo,ad[250,1:101],type='l',col="orange")
lines(eixo,ad[500,1:101],type='l',col="green")
lines(eixo,ad[750,1:101],type='l',col="blue")
lines(eixo,ad[1000,1:101],type='l',col="black")
#
# End of the second example
#
#
# Begin of the third example
# zero initial condition and a source.
# with advection, bi-blux (primary and secondary diffusion) and constant beta.
#
# Beta function
fbeta <- function(p)
{f <- 0.2
  return(f)}
# Beta derivative function
dbetadp <- function(p)
{f <- 0
  return(f)}

# Initial condition

```

```
fn <- function(x)
{ f <- 0
return(f)}

# velocity
v <- 0.00

# Source function
fs <- function(x,t)
{ if(x<=0.1){f <- 1}
  else{f <- 0}
return(f)}

# diffusion coefficients parameter
k2 <- 1e-3
k4 <- 1e-5

# Space and temporal definition
l <- 1
m <- 100
tf <- 1
n <- 1000

# Left boundary conditions
w10 <- 0
w11 <- 1
w12 <- 0
w20 <- 0
w21 <- 0
w22 <- 1
fw1 <- function(t)
{ f <- 0
  return(f)}
fw2 <- function(t)
{ f <- 0
  return(f)}

# Right boundary conditions
e10 <- 0
e11 <- 1
e12 <- 0
e20 <- 0
e21 <- 0
e22 <- 1
fe1 <- function(t)
{ f <- 0
  return(f)}
fe2 <- function(t)
{ f <- 0
  return(f)}
#
parm <- c(k2,k4,v,l,m,tf,n,w10,w11,w12,w20,w21,w22,e10,e11,e12,e20,e21,e22)
func <- c(fbeta=fbeta,dbetadp=dbetadp,fn=fn,fs=fs,fw1=fw1,fw2=fw2,fe1=fe1,fe2=fe2)
```

```

#
ad <- AdvDif4(parm,func)
eixo <- seq(0,1,by=0.01)
plot(eixo,ad[1000,1:101],type='l',col="black",xaxt="n",xlab="X", ylab="p(x,t)")
axis(1,seq(0,1,0.1),las=2)
lines(eixo,ad[250,1:101],type='l',col="orange")
lines(eixo,ad[500,1:101],type='l',col="green")
lines(eixo,ad[750,1:101],type='l',col="blue")
lines(eixo,ad[1,1:101],type='l',col="red")
#
# End of the third example
#
# It is easy to change k4 value in the previous example to observe its effect.
# Another possibility is to change beta function and its derivative also.
# There are more examples and also "News.md" inside "doc" folder.
#
#

```

---

pentaSolve

*Solving 'Ax=b' system*


---

## Description

This software solves an 'Ax=b' pentadiagonal system using a direct method. Variables a1, a2, a3, a4, a5 are matrix A diags and b is the vector.

## Usage

```
pentaSolve(a1, a2, a3, a4, a5, b)
```

## Arguments

a1	A vector
a2	A vector
a3	A vector
a4	A vector
a5	A vector
b	A vector

## Value

A vector with the x value

**Examples**

```
#  
# Solve a 'Ax=b' easy sample  
#  
a1<-c(1)  
a2<-c(2,2)  
a3<-c(7,7,7)  
a4<-c(2,2)  
a5<-c(1)  
b<-c(11,12,13)  
pentaSolve(a1,a2,a3,a4,a5,b)
```

# Index

AdvDif4, [2](#)

pentaSolve, [7](#)