Module ‘mininum’

Julio Manuel Fernández-Díaz

Profesor Titular, Department of Physics, University of Oviedo

February 2010

Table of contents

Module mininum

Basic information

`mininum` is a **mini**mal **num**erical library (with only three functions), developed for accompanying the `ask` helping system. The purpose is only to serve as an example.

`mininum` displays error messages in stderr.

Note: this library is very simple, and it is not intended for heavy calculation (but it is usable).

Call `ask"<function>"` for information on `<function>`.

List of functions

derivative quadrature root

Description of functions

mininum.derivative

Basic information

Calculates the first derivative of a function

Usage of function

```
mininum.derivative(f, x, aerr)
```

@params:

1. `f`: a function of a real variable.
2. `x`: number, is the abscissa at which *f'(x)* is calculated.
3. `aerr`: number (optional), is the intended absolute error of the solution (1.0e-6 as default). The minimum value of `aerr` is 1.0e-15.

@returns: number, the first derivative of function *f* at *x*.

More specific information

Function `mininum.derivative` uses the central difference formula:

f'(x0) ~ [f(x0+h)-f(x0-h)] / (2 h)

with successive decreasing values of *h: h/2, h/4, h/8*, etc., and applying the Richardson extrapolation method to improve the convergence.

See also

Press et al. (1992), Numerical Recipes in Fortran, p. 180, CUP.

Examples

```
require"mininum"

local eval = 0
local function g (x)
   eval = eval+1
   return sin(x)
end

local aerr = 1.e-6
local d1 = mininum.derivative(g, 2, aerr)
local e1 = cos(2)

print("calculated solution = ", d1)    -- -0.41614683654713
print("actual solution     = ", e1)    -- -0.41614683654714
print("intended error      = ", aerr)  -- 1e-06
print("actual absol. error = ", d1-e1) -- 9.9364960703952e-15
print("function evaluations= ", eval)  -- 10
```

mininum.quadrature

Basic information

Calculates the definite integral of a function

Usage of function

`mininum.quadrature(f, a, b, rerr)`

@params:

1. `f`: a function of a real variable.
2. `a`: number, the lower limit in the integral.
3. `b`: number, the upper limit in the integral.
4. `rerr`: number (optional), the relative intended error in the solution (if not given 1.0e-6 is assumed). The minimum value of `rerr` is 1.0e-15.

@returns: number, the definite integral of `f` between abscissas `a` and `b`.

More specific information

Function `mininum.quadrature` uses the **midpoint formula**:

I ~ h[sum f(xi)] with i = 1/2, 3/2, ...*

being n the number of intervals, $h = (b-a)/n$, and $xi = a+h*i$.

This formula works even for quadratures when the function at one or both limits is infinite but the integral exists.

The method is iterative, multiplying *n* by 3 at each step, until the relative error is achieved or a maximum of 14 iterations are reached (1594323 function evaluations). At least 9 ordinates are calculated.

At each iteration an *Aitken-delta^2* process is performed. This normally accelerates very much the convergence.

See also

Press et al. (1992), Numerical Recipes in Fortran, p. 129 and p. 160, CUP.

For the Aitken acceleration see the Wikipedia.

Examples

```
require"mininum"

local eval
local function s (x)
   eval = eval+1
   -- a difficult case for Romberg quadrature
   return 1/sqrt(x)
end

local rerr = 1.0e-5
local eval = 0
local q = mininum.quadrature(s, 0, 0.5, rerr)
local e = 2*sqrt(0.5)

print("calculated solution = ", q)       -- 1.414213736863
print("actual solution     = ", e)       -- 1.4142135623731
print("intended error      = ", rerr)    -- 1e-05
print("actual relat. error = ", (q-e)/e) -- 1.2338296867489e-07
print("function evaluations= ", eval)    -- 243
```

mininum.root

Basic information

Determines a root of a function between two abscissas

Usage of function

```
mininum.root(f, a, b, errx)
```

@params:

1. `f`: function of a real variable, of which we want the root, *x* such that *f(x) = 0*.
2. `a`: number.
3. `b`: number. The root is searched between the abscissas `a` and `b`.
4. `errx`: number (optional). The function returns a value when the difference between two successive approximations of the root is less than `errx` (in absolute value). If no value is provided for `errx`, then 1.0e-6 is assumed. The minimum value of `errx` is 1.0e-15.

@returns: number, an estimation of the root.

More specific information

function `mininum.root` uses **regula falsi** method.

The function must be continuous, and the provided abscissas *a* and *b* must accomplish *f(a)*f(b) < 0*. In this case the method always converge towards a solution.

It is an iterative method, with (somewhat) superlinear convergence. As much 30 iterations are done.

Here the "Illinois" version of the method is used.

See also

Wikipedia.

Examples

```
require"mininum"

local eval = 0
local function f (x)
   eval = eval+1
   return x*(3+x*(-4+2*x)) -- difficult for classical regula falsi
end

local errx = 1.0e-8
local x = mininum.root(f, -1, 1, errx)

print("calculated solution = ", x)     -- 3.9008079929199e-19
print("actual solution     = ", 0)     -- 0
print("intended error      = ", errx)  -- 1e-08
print("actual error        = ", x)     -- 3.9008079929199e-19
print("function evaluations= ", eval)  -- 14
```

Version

by Julio M. Fernández-Díaz, Dept. of Physics, University of Oviedo, Spain, Version 0.1, February 2010

julio a t uniovi d o t es

Notes