```
/
**************************************************************
************************/
/**
\n**/
/**                  n  e  w  t  o  n  .  c
\n**/
/**
\n**/
/**     Finds a zero of a function using the Newton's method
\n**/
/**
\n**/
/** (C) Potsdam Institute for Climate Impact Research (PIK),
see COPYRIGHT file    \n**/
/** authors, and contributors see AUTHORS file
\n**/
/** This file is part of LPJmL and licensed under GNU AGPL
Version 3               \n**/
/** or later. See LICENSE file or go to http://www.gnu.org/
licenses/               \n**/
/** Contact: https://gitlab.pik-potsdam.de/lpjml
\n**/
/**
\n**/
/
**************************************************************
************************/

#include <stdio.h>
#include <math.h>
#include "types.h"   /* Definition of datatype Real  */
#include "errmsg.h"
#include "numeric.h"

//#define DEBUG

Real newton(void (*fcn)(Real [2],Real,void *), /**< function
*/
            Real xstart,    /**< start value */
            void *data, /**< pointer to additional data for
function */
            Real xacc,   /**< accuracy in x */
            Real yacc,   /**< accuracy in y */
            int maxiter, /**< maximum number of iterations */
            int *iter,   /**< iterations actually performed */
            Bool *err    /**< error occured in function */
           )            /** \return position of zero of
function */
{
```

```
  Real x0,x1,f[2],dist;
#ifdef DEBUG
  Real df[2];
#endif
  int i;
  *iter=0;
  x0 = xstart+2.0*xacc;
  x1 = xstart;
  (*fcn)(f,x0,data);   /* f[0]=f(x0), f[1]=f'(x0) */
#ifdef DEBUG
  printf("0,x0=%g,f(x0)=%g,df(x0)=%g\n",x0,f[0],f[1]);
#endif
  for(i=0;i<maxiter;i++)
  {
    if (fabs(x1-x0) > xacc && fabs(f[0])>yacc)
    {
      dist=fabs(x1-x0);
      (*iter)++;
      x0=x1;
      (*fcn)(f,x0,data);   /* f[0]=f(x0), f[1]=f'(x0) */
#ifdef DEBUG
      (*fcn)(df,x0+1e-9,data);   /* f[0]=f(x0), f[1]=f'(x0) */
      printf("%d,
dx=%g,x0=%g,f(x0)=%g,df(x0)=%g,delta=%g\n",i+1,dist,x0,f[0],f[
1],(df[0]-f[0])/1e-9);
#endif
      if (fabs(f[1])==0.0)
      {
        *err=TRUE;
#ifdef DEBUG
        printf("error\n");
#endif
        return x1;
      }
      x1=x0-f[0]/f[1];
#if 1
      if(x1<0)
        x1=0;
#endif
    } /* of for */
    else
      break;
  }
  *err=FALSE;
  return x1;
} /* of 'newton' */
```