```c
/*
 * $Id: calch.c,v 1.24 2005/04/01 20:32:20 spoel Exp $
 *
 *                This source code is part of
 *
 *                 G   R   O   M   A   C   S
 *
 *          GROningen MAchine for Chemical Simulations
 *
 *                        VERSION 3.2.0
 * Written by David van der Spoel, Erik Lindahl, Berk Hess, and others.
 * Copyright (c) 1991-2000, University of Groningen, The Netherlands.
 * Copyright (c) 2001-2004, The GROMACS development team,
 * check out http://www.gromacs.org for more information.

 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version 2
 * of the License, or (at your option) any later version.
 *
 * If you want to redistribute modifications, please consider that
 * scientific software is very special. Version control is crucial -
 * bugs must be traceable. We will be happy to consider code for
 * inclusion in the official distribution, but derived work must not
 * be called official GROMACS. Details are found in the README & COPYING
 * files - if they are missing, get the official version at www.gromacs.org.
 *
 * To help us fund GROMACS development, we humbly ask that you cite
 * the papers on the package - you can find them in the top README file.
 *
 * For more info, check our website at http://www.gromacs.org
 *
 * And Hey:
 * GROningen Mixture of Alchemy and Childrens' Stories
 */
#ifdef HAVE_CONFIG_H
#include <config.h>
#endif

#include "macros.h"
#include "calch.h"
#include "maths.h"
#include "vec.h"
#include "physics.h"

#define xAI xa[0]
#define xAJ xa[1]
#define xAK xa[2]
#define xAL xa[3]
#define xH1 xh[0]
#define xH2 xh[1]
#define xH3 xh[2]
#define xH4 xh[3]

static void gen_waterhydrogen(int nh,rvec xa[], rvec xh[])
{
#define AA 0.081649
#define BB 0.0
#define CC 0.0577350
  const  rvec   matrix1[6] = {
    { AA,     BB,     CC },
    { AA,     BB,     CC },
    { AA,     BB,     CC },
    { -AA,    BB,     CC },
    { -AA,    BB,     CC },
    { BB,     AA,    -CC }
  };
  const  rvec   matrix2[6] = {
    { -AA,   BB,   CC },
```

```c
    { BB,    AA,  -CC },
    { BB,   -AA,  -CC },
    { BB,    AA,  -CC },
    { BB,   -AA,  -CC },
    { BB,   -AA,  -CC }
  };
#undef AA
#undef BB
#undef CC
  static int l=0;
  int        m;
  rvec       kkk;

  /* This was copied from Gromos */
  for(m=0; (m<DIM); m++) {
    xH1[m]=xAI[m]+matrix1[l][m];
    xH2[m]=xAI[m]+matrix2[l][m];
  }
  if (nh > 2)
    copy_rvec(xAI,xH3);
  if (nh > 3)
    copy_rvec(xAI,xH4);

  l=(l+1) % 6;
}

void calc_h_pos(int nht, rvec xa[], rvec xh[])
{
#define alfaH   (acos(-1/3.0)) /* 109.47 degrees */
#define alfaHpl (2*M_PI/3)     /* 120 degrees */
#define distH   0.1

#define alfaCOM (DEG2RAD*117)
#define alfaCO  (DEG2RAD*121)
#define alfaCOA (DEG2RAD*115)

#define distO   0.123
#define distOA  0.125
#define distOM  0.136

  rvec sa,sb,sij;
  real s6,rij,ra,rb,xd;
  int  d;

  s6=0.5*sqrt(3.e0);

  /* common work for constructing one, two or three dihedral hydrogens */
  switch (nht) {
  case 2:
  case 3:
  case 4:
  case 8:
  case 9:
    rij = 0.e0;
    for(d=0; (d<DIM); d++) {
      xd     = xAJ[d];
      sij[d] = xAI[d]-xd;
      sb[d]  = xd-xAK[d];
      rij   += sqr(sij[d]);
    }
    rij = sqrt(rij);
    sa[XX] = sij[YY]*sb[ZZ]-sij[ZZ]*sb[YY];
    sa[YY] = sij[ZZ]*sb[XX]-sij[XX]*sb[ZZ];
    sa[ZZ] = sij[XX]*sb[YY]-sij[YY]*sb[XX];
    ra = 0.e0;
    for(d=0; (d<DIM); d++) {
      sij[d] = sij[d]/rij;
      ra    += sqr(sa[d]);
    }
    ra = sqrt(ra);
    for(d=0; (d<DIM); d++)
      sa[d] = sa[d]/ra;

    sb[XX] = sa[YY]*sij[ZZ]-sa[ZZ]*sij[YY];
    sb[YY] = sa[ZZ]*sij[XX]-sa[XX]*sij[ZZ];
    sb[ZZ] = sa[XX]*sij[YY]-sa[YY]*sij[XX];
    break;
  }/* end switch */

  switch (nht) {
  case 1: /* construct one planar hydrogen (peptide,rings) */
    rij = 0.e0;
    rb  = 0.e0;
    for(d=0; (d<DIM); d++) {
      sij[d] = xAI[d]-xAJ[d];
      sb[d]  = xAI[d]-xAK[d];
      rij   += sqr(sij[d]);
      rb    += sqr(sb[d]);
    }
    rij = sqrt(rij);
    rb  = sqrt(rb);
    ra  = 0.e0;
    for(d=0; (d<DIM); d++) {
      sa[d] = sij[d]/rij+sb[d]/rb;
      ra   += sqr(sa[d]);
    }
    ra = sqrt(ra);
    for(d=0; (d<DIM); d++)
      xH1[d] = xAI[d]+distH*sa[d]/ra;
    break;
  case 2: /* one single hydrogen, e.g. hydroxyl */
    for(d=0; (d<DIM); d++) {
      xH1[d] = xAI[d]+distH*sin(alfaH)*sb[d]-distH*cos(alfaH)*sij[d];
    }
    break;
  case 3: /* two planar hydrogens, e.g. -NH2 */
    for(d=0; (d<DIM); d++) {
      xH1[d] = xAI[d]-distH*sin(alfaHpl)*sb[d]-distH*cos(alfaHpl)*sij[d];
      xH2[d] = xAI[d]+distH*sin(alfaHpl)*sb[d]-distH*cos(alfaHpl)*sij[d];
    }
    break;
  case 4: /* two or three tetrahedral hydrogens, e.g. -CH3 */
    for(d=0; (d<DIM); d++) {
      xH1[d] = xAI[d]+distH*sin(alfaH)*sb[d]-distH*cos(alfaH)*sij[d];
      xH2[d] = ( xAI[d]
                 - distH*sin(alfaH)*0.5*sb[d]
                 + distH*sin(alfaH)*s6*sa[d]
                 - distH*cos(alfaH)*sij[d] );
      if ( xH3[XX]!=NOTSET && xH3[YY]!=NOTSET && xH3[ZZ]!=NOTSET )
        xH3[d] = ( xAI[d]
                   - distH*sin(alfaH)*0.5*sb[d]
                   - distH*sin(alfaH)*s6*sa[d]
                   - distH*cos(alfaH)*sij[d] );
    }
    break;
  case 5: { /* one tetrahedral hydrogen, e.g. C3CH */
    real center;
    rvec dxc;

    for(d=0; (d<DIM); d++) {
      center=(xAJ[d]+xAK[d]+xAL[d])/3.0;
      dxc[d]=xAI[d]-center;
    }
    center=norm(dxc);
    for(d=0; (d<DIM); d++)
      xH1[d]=xAI[d]+dxc[d]*distH/center;
    break;
  }
```

```
  case 6: { /* two tetrahedral hydrogens, e.g. C-CH2-C */
    rvec rBB,rCC1,rCC2,rNN;
    real bb,nn;

    for(d=0; (d<DIM); d++)
      rBB[d]=xAI[d]-0.5*(xAJ[d]+xAK[d]);
    bb=norm(rBB);

    rvec_sub(xAI,xAJ,rCC1);
    rvec_sub(xAI,xAK,rCC2);
    oprod(rCC1,rCC2,rNN);
    nn=norm(rNN);

    for(d=0; (d<DIM); d++) {
      xH1[d]=xAI[d]+distH*(cos(alfaH/2.0)*rBB[d]/bb+
                           sin(alfaH/2.0)*rNN[d]/nn);
      xH2[d]=xAI[d]+distH*(cos(alfaH/2.0)*rBB[d]/bb-
                           sin(alfaH/2.0)*rNN[d]/nn);
    }
    break;
  }
  case 7:  /* two water hydrogens */
    gen_waterhydrogen(2, xa, xh);
    break;
  case 10: /* three water hydrogens */
    gen_waterhydrogen(3, xa, xh);
    break;
  case 11: /* four water hydrogens */
    gen_waterhydrogen(4, xa, xh);
    break;
  case 8: /* two carboxyl oxygens, -COO- */
    for(d=0; (d<DIM); d++) {
      xH1[d] = xAI[d]-distOM*sin(alfaCOM)*sb[d]-distOM*cos(alfaCOM)*sij[d];
      xH2[d] = xAI[d]+distOM*sin(alfaCOM)*sb[d]-distOM*cos(alfaCOM)*sij[d];
    }
    break;
  case 9: { /* carboxyl oxygens and hydrogen, -COOH */
    rvec xa2[4]; /* i,j,k,l   */

    /* first add two oxygens */
    for(d=0; (d<DIM); d++) {
      xH1[d] = xAI[d]-distO *sin(alfaCO )*sb[d]-distO *cos(alfaCO )*sij[d];
      xH2[d] = xAI[d]+distOA*sin(alfaCOA)*sb[d]-distOA*cos(alfaCOA)*sij[d];
    }

    /* now use rule 2 to add hydrogen to 2nd oxygen */
    copy_rvec(xH2, xa2[0]); /* new i = n' */
    copy_rvec(xAI, xa2[1]); /* new j = i  */
    copy_rvec(xAJ, xa2[2]); /* new k = j  */
    copy_rvec(xAK, xa2[3]); /* new l = k, not used */
    calc_h_pos(2, xa2, (xh+2));

    break;
  }
  default:
    gmx_fatal(FARGS,"Invalid argument (%d) for nht in routine genh\n",nht);
  } /* end switch */
}
```