user_function_m_n.gce

Attributes

```
  mainvars: x1 x2 ... y1 y2 ...
  iparms:
+   iprm1=0
+   iprm2=0
+   index_function=1
  rparms:
+   rprm1=0
+   rprm2=0
+   rprm3=0
+   rprm4=0
+   rprm5=0
+   rprm6=0
+   rprm7=0
+   rprm8=0
+   rprm9=0
+   rprm10=0
```

Description

`user_function_m_n.gce` provides a facility to the user to create a new function with `m` inputs and `n` outputs. Up to two integer parameters (`iprm1` and `iprm2`) and up to ten real parameters (`rprm1` to `rprm10`) can be used. The integer parameter `index_function` is passed on to the routine `user_function` as `n_given` (see the listing below), and it determines which function in `user_f.cpp` corresponds to the element being called.

To define a new function, the following steps need to be performed;

1. Edit `user_f.cpp` (located in the directory `sequel_fixed` in the linux distribution and in `SequelFixed` in the windows distribution) to include the new function.

   Note that the compile/link steps given below should be carried out in the directory where `user_f.cpp` is located.

2. Make sure that the `GNU g++` complier is installed on your computer.

3. Compile `user_f.cpp`:

   ```
   g++ user_f.cpp -o user_f.o
   ```

4. Link `user_f.o` with the SEQUEL object file to create the executable file:

   `g++ -static -O3 ccmain1u_fixed.o user_f.o -o ccmain_fixed` (Linux)

   `g++ -static -Wl,--stack,10000000 sqlmain1_cpp.o user_f.o -o sqlmain_cpp.exe` (Windows)

   (If there are compilation errors, repeat the above steps.)

In the following, the file `user_f.cpp` is reproduced (partially) to illustrate how the user's functions can be written and called.

```
#include <iostream>
#include <string>
#include <fstream>
#include <cstdlib>
#include <math.h>

#include "user_f1.h"

using namespace std;

.......
.......

void user_f_slip(
   double* x,
   double* y,
   int* iparm,
   double* rparm) {

// example of 3-input, 1-output function

   int poles;
   double tr;
   double w2,isq,imr,wrm,wmr;

   poles = iparm[0];
   tr    = rparm[0];

   isq = x[0];
   imr = x[1];
   wrm = x[2];

   if (fabs(imr) < 1.0e-2) {
     w2 = 1.0;
   } else {
```

```
      w2 = isq/(tr*imr);
   }

   wmr = 0.5*((double)(poles))*wrm + w2;

   y[0] = wmr;
   return;
}

void user_f_mux_1(
   const double time0,
   double* x,
   double* y,
   int* iparm,
   double* rparm) {

   double x1,x2,t0;

// example of 1-input, 2-output function

   x1 = x[0];
   x2 = x[1];
   t0 = rparm[0];

   if (time0 <= t0) {
     y[0] = x1;
     y[1] = 0.0;
   } else {
     y[0] = 0.0;
     y[1] = x2;
   }

   return;
}

void user_function(
   const int n_given,
   const double time0,
   double* x,
   double* y,
   int* iparm,
   double* rparm) {

   switch(n_given)
   {
   case 1 :
      user_f_1(x,y,iparm,rparm);
      break;
   case 2 :
```

```
      user_f_eff_d(x,y,iparm,rparm);
      break;
   case 3 :
      user_f_eff_q(x,y,iparm,rparm);
      break;
   case 4 :
      user_f_slip(x,y,iparm,rparm);
      break;
   case 5 :
      user_f_mux_1(time0,x,y,iparm,rparm);
      break;
   case 6 :
      cout << "user_function: n_given=6 is not implemented. Halting.." << endl; exit (1);
      break;
   case 7 :
      cout << "user_function: n_given=7 is not implemented. Halting.." << endl; exit (1);
      break;
   ........
   ........

   default :
      cout << "user_function: Check n_given. Halting..." << endl;
      exit (1);
   }
   return;
}
```

AC behaviour is not implemented.