

Solving Systems of Nonlinear Equations

This program (demosnle) demonstrates how to interact with the *Numerit* function snle which solves a system of user-defined nonlinear equations. The algorithm used in this numerical method is described in the book *Numerical Methods for Unconstrained Optimization and Nonlinear Equations* by J. E. Dennis, Jr. and R. B. Schnabel.

The problem solved by this example involves the conversion of geocentric coordinates to geodetic coordinates. The mathematical relationship between geocentric declination \mathbf{d} , geocentric distance r , geodetic latitude \mathbf{f} and geodetic altitude h is given by the following system of two nonlinear equations:

$$\begin{aligned}(c + h)\cos \mathbf{f} - r\cos \mathbf{d} &= 0 \\ (s + h)\sin \mathbf{f} - r\sin \mathbf{d} &= 0\end{aligned}\tag{1}$$

where

$$c = \frac{r_e}{\sqrt{1 - (2f - f^2)\sin^2 \mathbf{f}}}$$
$$s = c(1 - f)^2$$

In these equations r_e is the equatorial radius of the Earth and f is called the *flattening factor* of the Earth. The flattening factor is an indication of the *oblateness* of a planet.

The syntax and arguments for the snle function are given by

```
function snle(n, x, fvec, tol, info)
` solution of a systems of nonlinear equations
` input
` n = number of equations in nonlinear system
` x = initial guess for solution vector
` tol = solution tolerance
` output
` x = final solution vector
` fvec = nonlinear system evaluated at x
` info = solution information flag
` = 1 ==> improper input parameters
` = 2 ==> number of calls to userfunc has reached
` or exceeded 200 * (n + 1)
` = 3 ==> tol is too small. no further improvement
` in the approximate solution x is possible
` = 4 ==> iteration is not making good progress
```

Numerical Analysis with Numerit

The `snle` function requires user-coded function which contains the source code for the problem to be solved. The format of this function is as

```
function name(n, x, fvec, iflag)
` user-defined system of nonlinear equations
` input
` n = number of equations in system
` x = current argument vector
` output
` fvec = system of nonlinear equations evaluated at x
` iflag = indicator flag
` < 0 ==> user termination
` >= 0 ==> no action
```

where `name` is the actual name of the user-coded function. The function parameter list should be as defined here.

The main program uses "rerouting" to tell the `snle` function which function to use when calculating a solution. For this example the code is

```
snlefunc -> geosnle
```

The following is a source code listing of the user-defined system of nonlinear equations for this example.

```
function geosnle(n, x, fvec, iflag)
` user-defined system of nonlinear equations
` input
` n = number of equations in system
` x = current argument vector
` output
` fvec = system of nonlinear equations evaluated at x
` iflag = indicator flag
` < 0 ==> user termination
` >= 0 ==> no action
` Numerical Analysis with Numerit
.....
` "unload" current arguments

lat = x[1]
alt = x[2]
```

Numerical Analysis with Numerit

```
` compute geodetic factors

cfactor = %req / sqrt(1 - (2 * %flat - %flat^2) * sin(lat) *
                    sin(lat))

sfactor = cfactor * (1 - %flat)^2

` evaluate system of nonlinear equations

fvec[1] = (cfactor + alt) * cos(lat) - %rmag * cos(%dec)

fvec[2] = (sfactor + alt) * sin(lat) - %rmag * sin(%dec)
```

Notice how the Earth's equatorial radius `%req` and flattening factor `%flat` are passed to this function via global constants. The `iflag` argument in this function allows the user to terminate the iteration in case of a problem evaluating any of the components of the nonlinear system, etc.

The program will prompt you for the geocentric declination in degrees and the geocentric distance in kilometers. The declination can be any number between $\pm 90^\circ$ and the geocentric distance should be a positive number larger than the equatorial radius. The `snle` function requires an initial guess for each variable. This program uses the geocentric declination as the initial guess for the geodetic latitude and the difference between the geocentric distance and equatorial radius as the initial guess for the altitude.

The following is a typical example output created with this program.

```
function snle - system of nonlinear equations

solution

geodetic latitude (deg) = 45.18025787925662

geodetic altitude (km) = 429.7429698080186

nonlinear system at solution

fvec[1] = 0
fvec[2] = 0

solution information

info = 1

solution tolerance

tol = 1e-10
```

The convergence criterion is specified in the source code statement `tol = 1e-10` and can easily be changed by the user.