## NAME

prxma − CUTEr PRAXIS test driver

## SYNOPSIS

prxma

## DESCRIPTION

The *prxma* main program test drives PRAXIS on SIF problems from the CUTEr distribution.

## INTRODUCTION

The best introduction to PRAXIS is probably to quote the text (praxis.txt) distributed by J. Chandler, its initiator. Here is is.

Brent's PRAXIS minimizer is available in FORTRAN 77. July 1995

"Algorithms for Minimization Without Derivatives" by Richard P. Brent, Prentice-Hall, 1973 ISBN: 0-13-022335-2

This book by Brent was a groundbreaking effort. (I believe that it was his Ph.D. thesis at Stanford.) His algorithms for finding roots and minima in one dimension have good performance for typical problems and guaranteed performance in the worst case. (A later rootfinder by J. Bus and Dekker gave a much lower bound for the worst case, but no better performance in typical problems.) These algorithms were implemented in both ALGOL W and FORTRAN by Brent, and have been used fairly widely.

Brent also gave a multi-dimensional minimization algorithm, PRAXIS, but only shows an implementation in ALGOL W. This routine has not been widely used, at least in the U.S. The PRAXIS package has been translated into FORTRAN by Rosalee Taylor, Sue Pinski, and me, and I am making it available via anonymous ftp for use as freeware (please do not remove our names).

```
  ftp a.cs.okstate.edu
  anonymous
  [enter your userid as password]
  cd /pub/jpc
  get praxis.f
  quit
```

Brent's method and its performance Newton's method for minimization can find the minimum of a quadratic function in one iteration, but is sometimes not convenient to use. In the 1960s, several researchers found iterative methods that solve quadratic problems exactly in a finite number of steps. C. S. Smith (1962) and M. J. D. Powell (1964) devised methods that had this property and did not require derivatives. G. W. Stewart modified the Davidon-Fletcher-Powell quasi-Newton method to use finite difference approximations to approximate the gradient. Powell's method, or later versions by Zangwill, were the most successful of the early direct search methods having the property of finite convergence on quadratic functions.

Powell's method was programmed at Harwell as subroutine VA04A, and is available as file va04a.f in the same directory as praxis.f. VA04A is not extremely robust, and can give underflow, overflow, or division by zero. va04a.f has several documented patches in it where I tried to get around various abnormal terminations. I do not recommend VA04A very strongly.

Brent's PRAXIS added orthogonalization and several other features to Powell's method. Brent also dealt carefully with roundoff.

William H. Press et al. in their book "Numerical Recipes" comment that "Brent has a number of other cute tricks up his sleeve, and his modification of Powell's method is probably the best presently known."

Roger Fletcher was less enthusiastic in his review of Brent's book in The Computer Journal 16 (1973) 314: method are the best. Use of eigenvector directions is not independent of scale changes to the variables, and the use of searches in random directions is hardly appealing. Nonetheless all the algorithms are demonstrated to be competitive by numerical examples.'

The methods of Powell, Brent, et al. require that the function for which a local minimum is sought must be smooth; that is, the function and all of its first partial derivatives must be continuous.

Brent compared his method to the methods of Powell, of Stewart, and of Davies, Swann, and Campey. Indirectly, he compared it also to the Davidon-Fletcher-Powell quasi-Newton method. He found that his method was about as efficient as the best of these in most cases, and that it was more robust than others in some cases. (Pages 139-155 in Brent's book give fair comparisons to other methods. The results in Table 7.1 on page 138 are correct, but do not include progress all the way to convergence, and are therefore not too useful.)

On least squares problems, all of these general minimization methods are likely to be inefficient compared to least squares methods such as the Gauss-Newton or Marquardt methods.

In addition to the scale dependence that Fletcher deplored, PRAXIS also had the disadvantage that it required N, the number of parameters, to be greater than or equal to two. The failure to handle N=1 is an unnecessary and pointless limitation.

## The FORTRAN version

We have followed Brent's PRAXIS rather closely. I have added a patch to try to handle the case N=1, and an option to use a simpler pseudorandom number generator, DRANDM. The handling of N=1 is not guaranteed.

The user writes a main program and a function subprogram to compute the function to be minimized. All communication between the user's main program and PRAXIS is done via COMMON, except for an EXTERNAL parameter giving the name of the function subprogram. The disadvantages of using COMMON are at least two-fold:

1) Arrays cannot have adjustable dimensions.

2) Because some actual parameters are COMMON variables, the FORTRAN version of PRAXIS probably will not pass the Bell Labs PFORT package as being 100% standard FORTRAN. Nevertheless, this usage will not cause any conflict in any commercial FORTRAN compiler ever written. (If it does, I will apologize and rewrite PRAXIS.)

The advantage of using COMMON is that it is not necessary to pass about fifteen more parameters every time the user calls PRAXIS. At present all arrays are dimensioned (20) or (20,20), and this can easily be increased using two simple global editing commands. (In this case, increase the value of NMAX.)

There are no DATA statements in PRAXIS, and it was not necessary to use any SAVE statements.

We have used DOUBLE PRECISION for all floating point computations, as Brent did. We recommend using DOUBLE PRECISION on all computers except possibly Cray computers, in which

REAL is reasonably precise. The value of "machine epsilon" is computed in subroutine PRASET using bisection, and is called EPSMCH. Brent computes EPSMCH**4 and 1/EPSMCH**4 in PRAXIS, and uses these quantities later. Because EPSMCH in DOUBLE PRECISION is less than 1E-16, these fourth powers of EPSMCH and 1/EPSMCH will underflow and overflow on such machines as VAXs and PCs, which have a range of only about 1E38, grossly insufficient for scientific computation. For such machines, Brent recommends increasing the value of EPSMCH. EPSMCH=1E-9 or possibly even 1E-8 might be necessary. A better solution would be to eliminate the explicit use of these fourth powers, accomplishing the same result implicitly.

A "bug bounty" of $10 U.S. will be paid by me for the first notification of any error in PRAXIS. The same bounty also applies to any substantive poor design choice (having no redeeming advantages whatever) in the FORTRAN package. (The patch for N=1 is not included, although any suggested improvements in that will be considered carefully.)

praxis.f includes test software to run any of the test problems that Brent ran, and is set to run at least one case of each problem. I have run these on an IBM 3090, essentially the same architecture that Brent used, and obtained essentially the same results that Brent shows on pages 140-155. The Hilbert problem with N=12, for which Brent shows no termination results and for which the results in Table 7.1 are correct but not relevant, runs a long time; I cut it off at 3000 function evaluations. I don't particularly like Brent's convergence criterion, which allows this sort of extremely slow creeping progress, but have not modified it.

Please notify me of any problems with this software, or of any suggested modifications.

John Chandler Computer Science Department Oklahoma State University Stillwater, Oklahoma 74078, U.S.A. (405) 744-5676 jpc@a.cs.okstate.edu

Of course, the paragraph on test problems no longer applies when PRAXIS is used in conjunction with CUTE, but many more problems may then be tested!

## USAGE

The object module *prxma.o* is stored in $MYCUTER/*precision*/bin, where *precision* is either "single" or "double", according to your local installation.

Starting from the praxis.f file distributed by J. Chandler, perform the following steps.

1) Remove the driver program, the subroutines TESTIN, FTEST and PRASET. The
   first routine is then PRAXIS(F) itself.

2) Change the name of the subroutine RANDOM to PRXRDM, say, everywhere in the
   code. This is necessary because CUTE also uses a function called RANDOM (in
   the others.f file. To avoid multiply defined entries when PRAXIS is linked
   with the CUTE tools, the duplicate names must thus be removed.

3) At the beginning of the code, replace the section

```
C
C  MACHINE DEPENDENT NUMBERS...
C
C  ON MANY COMPUTERS, VSMALL WILL UNDERFLOW,
C  AND COMPUTING XLARGE MAY CAUSE A DIVISION BY ZERO.
C  IN THAT CASE, EPSMCH SHOULD BE SET EQUAL TO 1.0D-9
C  (OR POSSIBLY 1.0D-8) BEFORE CALLING PRAXIS.
C
```

```
      SMALL=EPSMCH*EPSMCH
      VSMALL=SMALL*SMALL
      XLARGE=ONE/SMALL
      VLARGE=ONE/VSMALL
      XM2=ZSQRT(EPSMCH)
      XM4=ZSQRT(XM2)
```

by

```
C
C  MACHINE DEPENDENT NUMBERS...
C
      VSMALL = 1.0D-35
      SMALL  = 1.0D-20
      VLARGE = 1.0D+35
      XLARGE = 1.0D+20
      XM2    = ZSQRT( EPSMCH )
      XM4    = ZSQRT( XM2)
```

4) Save the resulting Fortran code in a file named praxisd.f.

5) Compile (but do not link) praxisd.f. The resulting object file praxisd.o should be placed in $MYCUTER/*precision*/bin. Launch using prx(1) or sdprx(1).

## NOTE

If no PRX.SPC file is present in the current directory, the default version is copied from $CUTER/common/src/pkg/praxis/. The default specifications are as follows

```
100000        NFMAX,        maximum number of function calls
0.00001       T,            stopping tolerance
1.0           SCBD,         upper bound on the scale factors
0             ILLCIN,       "ill-conditioning" flag
1             KTM,          maximum number of iterations
                             without improvement
0             JPRINT,       printing specifier
```

The reader is referred to the paper quoted below and the code itself if they wish to modify these parameters.

## ENVIRONMENT

**CUTER**

Parent directory for CUTEr

**MYCUTER**

Home directory of the installed CUTEr distribution.

## AUTHORS

I. Bongartz, A.R. Conn, N.I.M. Gould, D. Orban and Ph.L. Toint

## SEE ALSO

*CUTEr (and SifDec): A Constrained and Unconstrained Testing Environment, revisited*,
N.I.M. Gould, D. Orban and Ph.L. Toint,
ACM TOMS, **29**:4, pp.373-394, 2003.

*CUTE: Constrained and Unconstrained Testing Environment*, I. Bongartz, A.R. Conn, N.I.M. Gould and Ph.L. Toint, TOMS, **21**:1, pp.123-160, 1995.

*Algorithms for Minimization Without Derivatives*, by Richard P. Brent, Prentice-Hall, 1973 ISBN: 0-13-022335-2