ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

User Guide for the MINPACK-2 Test Problem Collection

Brett M. Averick[†], and Jorge J. Moré

Mathematics and Computer Science Division

Technical Memorandum No. 157

October 1991

Work supported in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under Contract W-31-109-Eng-38 and in part by the Army Research Office under grant DAALO3-89-C-0038.

[†] Army High Performance Computing Research Center (AHPCRC), Institute of Technology, University of Minnesota, 1100 Washington Avenue South, Minneapolis, Minnesota 55415

User Guide for the MINPACK-2 Test Problem Collection

by

Brett M. Averick and Jorge J. Moré

ABSTRACT

The Army High Performance Computing Research Center at the University of Minnesota and the Mathematics and Computer Science Division at Argonne National Laboratory are collaborating on the development of the software package MINPACK-2. As part of the MINPACK-2 project we are developing a collection of significant optimization problems to serve as test problems for the package. This report describes the software associated with the preliminary version of the MINPACK-2 test problem collection. The discussion centers on the description of subroutine parameters, but additional information on the implementation of these subroutines is also provided. The information in this report should be sufficient to use these subroutines for testing and evaluating software.

1 Introduction

This report describes software associated with the preliminary version of the MINPACK-2 test problem collection. The discussion centers on the description of subroutine parameters, but additional information on the implementation of these subroutines is also provided. The information in this report should be sufficient to use these subroutines for testing and evaluating software.

Averick, Carter, and Moré [1] describe the preliminary version of the MINPACK-2 test problem collection. This reference should be consulted for information on problem formulation and for additional details on the relationship between the subroutine parameters and the original problem.

The problems contained in the current version of the test problem collection fall into three categories:

1. Systems of Nonlinear Equations.

2. Nonlinear Least Squares.

3. Minimization.

For systems of nonlinear equations and nonlinear least squares problems, code is supplied for the evaluation of the vector-valued functions and the Jacobian matrices; for minimization problems, code is supplied for the evaluation of the functions and the gradients. In addition, if lower or upper bounds are part of the problem formulation, code is supplied for the evaluation of the bounds.

These subroutines were written to be easily read and understood. More efficient implementations that take advantage of high-performance architectures are possible. Note that there is no checking of input parameters in these subroutines. The information provided in this report should be sufficient to use these subroutines safely, but the user should read the documentation within each subroutine for additional information. Particular attention should be paid to problem dimension parameters.

Code listings for the double precision version of these subroutines can be found in the appendix. Single precision versions of these subroutines are also available. Either version can be obtained by sending email to

`averick@mcs.anl.gov`

and specifying which version is desired. Comments on these subroutines or suggestions for additional problems can also be sent to this address.

2 Systems of Nonlinear Equations

The subroutines described in this section define systems of nonlinear equations of the form

$$f(x) = 0, \qquad x_l \le x \le x_u,$$

where $f : \Re^n \to \Re^n$ defines the residuals, and x_l, x_u are bounds on the solution. The action of the subroutines depends on the character variable `task` as follows:

Evaluate the function if `task = 'F'`.

Evaluate the Jacobian matrix if `task = 'J'`.

Evaluate the function and the Jacobian matrix if `task = 'FJ'`.

Evaluate the standard starting point x_s if `task = 'XS'`.

Evaluate the lower bound x_l if `task = 'XL'`.

Evaluate the upper bound x_u if `task = 'XU'`.

The function value is returned in the array `fvec`, the Jacobian matrix is returned in the array `fjac` (with leading dimension `ldfjac`), and the starting point x_s and bounds x_l, x_u are returned in the array `x`.

Problems 2.1 and 2.2 arise in the discretization of systems of boundary value problems in one spatial dimension by a k-stage collocation method. If boundary conditions are given at $t = a$ and $t = b$, and

$$a = t_1 < t_2 < \cdots < t_{n_0} < t_{n_0+1} = b$$

is a partitioning of $[a, b]$ into n_0 subintervals, the collocation method approximates the solution to a system of p boundary value problems by a vector-valued piecewise polynomial function $u_\pi : [a, b] \to \Re^p$. The use of the collocation method leads to a system of nonlinear equations. An advantage of the implementation of the collocation method used in these test problems is that if m_s is the order of the s-th boundary value problem, then the s-th component of

$$u_\pi^{(j-1)}(t_i), \qquad 1 \le i \le n_0, \quad 1 \le j \le m_s,$$

is directly available from the solution vector to the system of nonlinear equations. The s-th component of $u_\pi^{(j-1)}(t_i)$ is stored in the $\sigma_p(i-1) + \sigma_{s-1} + j$ location of the array `x`, where

$$\sigma_r = rk + \sum_{l=1}^{r} m_l.$$

This information is useful for obtaining plots of the j-th derivative of the s-th component of u_π. If the variable `nint` is the number n_0 of subintervals, and the variables `n1` and `n2` contain σ_p and σ_{s-1}, respectively, then the pseudo-code

```
do i = 1, nint
   v(i) = x((i-1)n1+n2+j)
end do
```

stores in the array `v` the value of the s-th component of $u_\pi^{(j-1)}(t_i)$. Given the array `v`, standard plotting subroutines can be used to obtain plots of $u_\pi^{(j-1)}$.

2.1 Flow in a Channel

The subroutine

```
dflow(n,x,fvec,fjac,ldfjac,task,r,nint)
```

defines the flow in a channel problem. The parameter `r` is the Reynolds number R and must be positive. The parameter `nint` is the number n_0 of subintervals in the collocation method used to discretize the boundary value problem. The user must provide a positive value of `nint` and set `n = 8*nint`.

The tangential velocity u' is of interest in this problem. Plots of the tangential velocity can be obtained by noting that $u'(t_i)$, $1 \le i \le n_0$, is stored in location $8(i-1)+2$ of the array `x`.

2.2 Swirling Flow

The subroutine

```
dswrl3(n,x,fvec,fjac,ldfjac,task,eps,nint)
```

defines the swirling flow problem. The parameter `eps` is the viscosity parameter ϵ and must be positive. The parameter `nint` is the number n_0 of subintervals in the collocation method used to discretize the boundary value problem. The user must provide a positive value of `nint` and set `n = 14*nint`.

The radial velocity f' and the angular velocity g are of interest in this problem. Plots of the radial velocity can be obtained by noting that $f'(t_i)$, $1 \le i \le n_0$, is stored in location $14(i-1)+2$ of the array `x`. Plots of the angular velocity can be obtained by noting that $g(t_i)$, $1 \le i \le n_0$, is stored in location $14(i-1)+9$ of the array `x`.

2.3 The Human Heart Dipole

The subroutine

```
ddgv(n,x,fvec,fjac,ldfjac,task,prob)
```

defines the human heart dipole problem. The user must set `n` = 8. The parameter `prob` specifies one of five versions of the problem. Lower and upper bounds are provided.

2.4 Combustion of Propane

The subroutine

```
dmorgf(n,x,fvec,fjac,ldfjac,task)
```

defines the combustion of propane problem. The user must set `n` = 11. Lower bounds are provided.

2.5 Combustion of Propane – Reduced Formulation

```
dmorgr(n,x,fvec,fjac,ldfjac,task)
```

defines the reduced formulation of the combustion of propane problem. The user must set `n` = 5. Lower bounds are provided.

3 Least Squares Problems

The subroutines described in this section define nonlinear least squares problems of the form

$$\min\{\|f(x)\|_2^2 : x_l \leq x \leq x_u\},$$

where $f : \Re^n \to \Re^m$ defines the residuals of the least squares problem, and x_l, x_u are bounds on the solution. The action of the subroutines depends on the character variable `task` as follows:

Evaluate the function if `task = 'F'`.

Evaluate the Jacobian matrix if `task = 'J'`.

Evaluate the function and the Jacobian matrix if `task = 'FJ'`.

Evaluate the standard starting point x_s if `task = 'XS'`.

Evaluate the lower bound x_l if `task = 'XL'`.

Evaluate the upper bound x_u if `task = 'XU'`.

The function value is returned in the array `f`, the Jacobian matrix is returned in the array `fjac` (with leading dimension `ldfjac`), and the starting point x_s and bounds x_l, x_u are returned in the array `x`.

3.1 Isomerization of α-pinene – Direct Formulation

The subroutine

```
dapdir(m,n,x,fvec,fjac,ldfjac,task,nh)
```

defines the direct formulation of the α-pinene problem. The user must set `m` = 40 and `n` = 5. The parameter `nh` is the number of consecutive Runge-Kutta steps taken between observations. A typical value is `nh` = 10. Lower bounds are provided.

3.2 Isomerization of α-pinene – Collocation Formulation

The subroutine

```
dapcol(m,n,x,fvec,fjac,ldfjac,task,nint,sigma)
```

defines the collocation formulation of the α-pinene problem. The parameter `nint` is the number of subintervals in the collocation method used to discretize the initial value problem. The user must provide a positive value of `nint` and set `m` = 25∗`nint`+40 and `n` = 25∗`nint`+5. The parameter `sigma` defines the weights used in the l_2 penalty approach by setting $\sigma_i = \sigma$, where σ_i is the weight for the i-th constraint, and σ is the value defined by `sigma`. A typical value is $\sigma = 10^6$.

As is the case for problems 2.1 and 2.2, the vector `x` contains information on the approximate solution to the original problem. In this problem the approximate solution $u(\cdot, \theta)$ to the linear kinetic problem is of interest. Plots of the components of $u(\cdot, \theta)$ can be obtained by noting that the s-th component of $u(t_i, \theta)$, $1 \leq i \leq n_0$, is stored in location $25(i-1) + 5(s-1) + 1$ of the array array `x`.

3.3 Coating Thickness Standardization

The subroutine

```
drog1(m,n,x,fvec,fjac,ldfjac,task)
```

defines the coating thickness standardization problem. The user must set `m` = 252 and `n` = 134.

3.4 Exponential Data Fitting I

The subroutine

```
dosb1(m,n,x,fvec,fjac,ldfjac,task)
```

defines the exponential data fitting I problem. The user must set `m` = 33 and `n` = 5. Lower and upper bounds are provided.

3.5 Exponential Data Fitting II

The subroutine

```
dosb2(m,n,x,fvec,fjac,ldfjac,task)
```

defines the exponential data fitting II problem. The user must set `m` = 65 and `n` = 11. Lower and upper bounds are provided.

3.6 Thermistor Resistance

The subroutine

```
dmeyer(m,n,x,fvec,fjac,ldfjac,task)
```

defines the thermistor resisitance problem. The user must set `m` = 16 and `n` = 3.

3.7 Analysis of an Enzyme Reaction

The subroutine

```
dkmor(m,n,x,fvec,fjac,ldfjac,task)
```

defines the analysis of an enzyme reaction problem. The user must set `m` = 11 and `n` = 4.

3.8 Chebyshev Quadrature

The subroutine

```
dcquad(m,n,x,fvec,fjac,ldfjac,task)
```

defines the Chebyshev quadrature problem. Any positive values of `m` $\geq$ `n` are permissible. Lower and upper bounds are provided.

4 Minimization Problems

The subroutines described in this section define minimization problems of the form

$$\min\{f(x) : x_l \leq x \leq x_u\},$$

where $f : \Re^n \rightarrow \Re$, and the vectors x_l, x_u specify bounds on the solution. The action of the subroutines depends on the character variable `task` as follows:

Evaluate the function if `task = 'F'`.

Evaluate the gradient if `task = 'G'`.

Evaluate the function and the gradient if `task = 'FG'`.

Evaluate the standard starting point x_s if `task = 'XS'`.

Evaluate the lower bound x_l if `task = 'XL'`.

Evaluate the upper bound x_u if `task = 'XU'`.

The function value is returned in the parameter `f`, the gradient is returned in the array `fgrad`, and the starting point x_s and bounds x_l, x_u are returned in the array `x`.

The problems in this section arise as finite element approximations to a variational problem. The vector $x \in \Re^n$ defines a piecewise linear approximation v to the solution of the variational problem. For these problems the approximation v is defined on a triangulation of a rectangular domain $\mathcal{D}$ with n_x interior points in the x-direction and n_y interior points in the y-direction. The value of v at the (i, j) vertex of the triangulation is stored in the $(j-1)n_x + i$ location of the array `x`. Thus, the pseudo-code segment

```
do j = 1, ny
   do i = 1, nx
      v(i,j) = x((j-1)nx+i)
   end do
end do
```

stores in the array `v` the values of the approximation v. Given the array `v`, standard plotting subroutines can be used to obtain plots of the approximation v. These problems can be made arbitrarily large by increasing n_x and n_y. In this formulation $n_y = 1$ for 1-dimensional domains.

4.1 Elastic-Plastic Torsion

The subroutine

```
dtor(nx,ny,x,f,fgrad,task,c)
```

defines the elastic-plastic torsion problem. The parameter `c` is the angle c of twist per unit length on the bar. A typical value is $c = 5$. Lower and upper bounds are provided.

4.2 Pressure Distribution in a Journal Bearing

The subroutine

```
djourb(nx,ny,x,f,fgrad,task,ecc,b)
```

defines the journal bearing problem. The parameter `ecc` is the eccentricity ϵ of the journal bearing, and `b` specifies the domain $\mathcal{D} = (0, 2\pi) \times (0, b)$. Typical values for these parameters are $\epsilon = 0.1$ and $b = 10$. Lower bounds are provided. The subroutine

```
djours(nx,ny,w,ecc,b)
```

determines a diagonal scaling matrix `w` as the square root of the diagonal elements of the Hessian matrix.

4.3 Minimal Surfaces

The subroutine

```
dmnsur(nx,ny,x,f,fgrad,task,bottom,top,left,right)
```

defines the minimal surface problem. The arrays `bottom,top,left,right` specify the boundary conditions for the surface. The subroutine

```
dennbc(nx,ny,hx,hy,bottom,top,left,right)
```

determines the boundary conditions for Enneper's minimal surface. Other boundary conditions can be obtained by modifying `dennbc`.

4.4 Optimal Design with Composite Materials

The subroutine

```
doptd(nx,ny,x,f,fgrad,task,lambda)
```

defines the optimal design problem. The parameter `lambda` is the multiplier λ of the optimal design problem. Values of interest are $\lambda \in [0, 1]$; a typical value is $\lambda = 0.008$ The subroutine

```
dpsif(t,mu1,mu2,t1,t2,result,task,lambda)
```

computes $\psi(t)$ and computes $\psi'(t)/t$. Other functions can be obtained by modifying `dpsif`.

4.5 Inhomogeneous Superconductors

The subroutine

```
dgland(n,x,f,fgrad,t)
```

defines the inhomogeneous superconductor problem. The parameter `t` specifies the temperature t. Values of interest are $t \in [3.73, 7.32]$; a typical value is $t = 5$.

Acknowledgments

We are grateful, in particular, to Richard Carter, David Gay, and Janet Rogers for their test problems. Many others contibuted with comments and suggestions for improvements. In particular, Uri Ascher, Karin Bennett, Larry Biegler, Burt Garbow, Jonathan Goodman, Bob Maier, Alex Morgan, and Layne Watson.

References

[1] B. Averick, R. Carter, and J. Moré, *The MINPACK-2 test problem collection (Preliminary Version)*, Technical Memorandum ANL/MCS-TM-150, Argonne National Laboratory, Argonne, Illinois, 1991.