Abstract: A subroutine for the log likelihood maximization is proposed. It is an implementation of Davidon’s variance algorithm with a numerical derivative evaluation procedure. It is designed for use of statistical model builders. It is originally coded in FORTRAN. Now C version is available. Numerical examples are given.

Key Words: Statistical modeling, log likelihood, AIC, Numerical optimization, Davidon’s variance algorithm, Numerical differentiation, MLE, FORTRAN, parallel computation, Open Market Licence

1 Introduction

By the introduction of AIC (Akaike, 1973) it is recognized that many important statistical problems could be formulated as model selection problems. Necessary statistical modeling is the task not only of expert statisticians, but of all those who concern with the data analysis. Here the log likelihood maximization is the key operation. Typical steps for the statistical problem solving are as follows:

1. model building
2. log likelihood maximization
3. AIC computation and model selection
4. interpretation of the results.

The hindrance there, if any, is the difficulty of numerical maximization of the likelihood. Numerical procedures should be used for the maximization. Traditional statistical procedures seem to be designed so that the computing cost will be as low as possible. However, taking into account the recent development of computing technology, we need not be afraid of some computation. We may relax our attachment to the analytical methods, so that we will be able to concentrate our attention more on the physical aspect of data.

The optimization procedure should be one that works without user supplied gradient evaluation algorithm. The run-time cost could be reduced by the gradient evaluation algorithm. However, it is usual that the necessary effort to write down the algorithm is substantial and it is not rare that the obtained algorithm contradicts the algorithm for the calculation of the function value. When the analysts is groping for a good model to describe a given phenomenon, the expectation of the possible trouble often suppresses the desire to develop some better ideas.
The purpose of the present article is to propose one working procedure which meets the above stated requirements. We do not claim that ours is the best of all the optimization techniques in the literature. We do not even try to compare the performances of methods. We just make public our procedure and encourage every data analyst to participate in the model building game. Those who are interested in other methods are referred to, for example, Douke & Asano (1989), Kowalik & Osborne (1968), Künzi et al. (1968), Lavi & Vogl (1966) and Powell (1981).

2 DALL

Read "DALL" so that it rhymes with "call". Following the above reasoning, we proposed, in 1996, a FORTRAN subroutine DALL for numerical maximization of the log likelihood function which was based on Davidon's algorithm (Davidon, 1968) and numerical differentiation. The choice of the Davidon's algorithm was based on the experience of the second author in the implementation of the maximum likelihood procedure for ARMA time series model in TIMSAC-74 (Akaike et al., 1985). The subroutine BILL for Davidon's algorithm is a FORTRAN77 version of the maximization procedure in TIMSAC-74. The construction and coding of DALL was mainly done by the first author. Now we propose a new FORTRAN version and an equivalent C version. The flow chart of DALL is given as Figure 1.

2.1 Davidon's Algorithm

Assume that a function $\phi$ has the form

$$
\phi(x) = \phi_0 + \frac{1}{2}(x - x_0)^T V_0^{-1}(x - x_0),
$$

(1)

where $V_0^{-1}$ is a negative definite matrix, then the relation between the gradient at a point $x$ and the maximizing point is given by

$$
g = V_0^{-1}(x - x_0).$$

(2)

That is, if the variance $V_0$ of the function (1) is known and the gradient at a point is known the maximizing point is readily calculated by

$$x - V_0g.$$  

(3)

On the other hand, if gradients at sufficient number of points are known $V_0$ can be estimated. Assuming that the object function is well approximated by a quadratic function, Davidon’s variance algorithm (Davidon, 1968), a typical quasi-Newton method, is constructed on these two facts.

Notes to Figure 1

(1) The gradient is computed numerically by

$$g_i = \frac{\phi(x + \text{step}(i)e_i) - \phi(x - \text{step}(i)e_i)}{2 \text{step}(i)} \quad (i = 1, 2, \ldots, IP)$$

(4)

where $e_i$ is the unit vector of the $i$-th axis and IP is the dimension of the parameter vector $\theta$. This computation can (and should, if possible) be executed parallely.

(2) The difference between $SPHI$ and the maximum value is given by

$$\frac{1}{2}(x_\ast - x_0)^T V_0^{-1}(x_\ast - x_0),$$

which is equal to

$$\frac{1}{2}g_\ast^TV_0g_\ast.$$  

(5)

$RHO$ is the twice of an estimate of this value. It is implicitly assumed that the variance $V_n$ is sufficiently close to $V_0$. 

2
Figure 1. Flow chart of DALL

(3) Under the present assumptions, if $V_n = V_0$ then $g_\ast = 0$ and the procedure ends. The fact that $g_\ast \neq 0$ is an evidence of $V_n \neq V_0$. If (1) is the correct form, $V_0$ satisfies

$$x_n - x_\ast = V_0(g_n - g_\ast).$$

(6)
Define $\gamma = -g^TV_n g_*/\rho$, then $\lambda_* = |\gamma/(\gamma+1)|$.

(4) If $\lambda$ = 1, $V_n$ is not improved. If $\lambda = 1$ is far from 1 the modification might be too drastic, especially if the assumed form (1) of the function is dubious.

The careful choice of $\alpha$ and $\beta$, which appear in step 3(c), is important. The choice is discussed by Davidson(1968) and $\alpha = 10^{-3}$ and $\beta = 10$ are suggested as reasonable choices. However, based on our own experience of the use of the procedure we set $\alpha$ and $\beta$ equal to 0.25 and 4.0, respectively.

(5) Generally, the shape of a $K$-dimensional matrix $A$ is known by calculating $\{Ab_1, Ab_2, Ab_3, ..., Ab_K\}$, where $\{b_1, b_2, b_3, ..., b_K\}$ is an orthogonal system of $K$ vectors. If we choose the basis so that $b_1 = V_n g_*$, we have relations

$$V_{n+1} b_j = V_n b_j \quad (j = 2, ..., K).$$

(7)

(6) Tis path is not a part of Davidson’s original algorithm.

2.2 DALL

Explanations hereafter are for the FORTRAN version. It should not be too difficult translate the following text into words of C.

Usage

Prepare

```c
subroutine model(a,ip,el,lel)
```
to compute the log likelihood, initial guess ‘a’ of parameter and step sizes step(1), step(2), ... , step(ip) for numerical differentiation, and

```c
call dall(model,a,ip,el,g vd,endc,step,limit,lpsw),
```
then the variable ‘el’ gives the maximum log likelihood. The specification for the make of ‘model’ is as follows:

- The number of free parameters is given in ‘ip’.
- Parameter values are set in ‘a’.
- Set log likelihood value in ‘el’.
- Set ‘lel’=0, if given ‘a’ is out of domain of the log likelihood. Otherwise set ‘lel’=1.

The meaning of other arguments are summarized in Table 1.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>model</td>
<td>input. subroutine name.</td>
</tr>
<tr>
<td>a(ip)</td>
<td>input/output. parameters.</td>
</tr>
<tr>
<td>ip</td>
<td>input. number of free parameter.</td>
</tr>
<tr>
<td>el</td>
<td>output. maximum log likelihood.</td>
</tr>
<tr>
<td>g(ip)</td>
<td>output. final gradient estimate.</td>
</tr>
<tr>
<td>vd(ip,ip+5)</td>
<td>output/work area.</td>
</tr>
<tr>
<td>endc</td>
<td>output. end code.</td>
</tr>
<tr>
<td>step(ip)</td>
<td>input. step size vector for the numerical differentiation.</td>
</tr>
<tr>
<td>limit</td>
<td>input. NCOUNT limit. limit=0 indicates no limit.</td>
</tr>
<tr>
<td>lpsw</td>
<td>input. output level control (see Table 3).</td>
</tr>
</tbody>
</table>
Outputs Arguments for outputs are picked up in Table 2. Table 3 summarizes output control by variable ‘lpsw’.

<table>
<thead>
<tr>
<th>Table 2 Results of DALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Argument</td>
</tr>
<tr>
<td>a(ip)</td>
</tr>
<tr>
<td>el</td>
</tr>
<tr>
<td>g(ip)</td>
</tr>
<tr>
<td>vd(ip,ip)</td>
</tr>
<tr>
<td>endc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 3 Output control</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpw</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

When ‘lpw’ is greater than 1, “global profile of the log likelihood” is given. This ‘graph’ shows the profile of the log likelihood along the bee-line from the initial point to the final point.

When ‘lpw’ is set equal to 0, user has to check ‘endc’ variable to know the status. ‘endc’ is defined by

\[ \text{endc} = \text{NCOUNT} + \text{LCODE}/100 \]

where LCODE has the meaning summarized in Table 4.

<table>
<thead>
<tr>
<th>Table 4 LCODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>lcode</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>50</td>
</tr>
<tr>
<td>70</td>
</tr>
<tr>
<td>71</td>
</tr>
<tr>
<td>72</td>
</tr>
<tr>
<td>85</td>
</tr>
<tr>
<td>90</td>
</tr>
<tr>
<td>95</td>
</tr>
</tbody>
</table>

3 Numerical Examples

3.1 One dimensional optimization

This example gives a side view of the performance of DALL. Assume that we conducted coin tossing \( n \) times and got \( m \) times “top”. Then the log likelihood function of the probability \( p \) of getting “top” is given by

\[
\ell(p) = m \log p + (n - m) \log(1 - p). \tag{8}
\]

Actually MLE is analytically calculated to be \( \hat{p} = m/n \).

The example is for the case \( n = 12, m = 76 \) and then \( \hat{p} = 0.158 \). We can see how Dall finds the maximum value staring from 0.9 as the initial guess in Fig.2. It is seen that quadratic function model

\[
\phi(x) = \phi^* + \frac{1}{2}(x - x^*)^TV_n^{-1}(x - x^*). \tag{9}
\]

fitted to the data is improved gradually. The position of the tangent point of the graph of \( \ell(p) \) and the graph of the model is \( x_{\text{NCOUNT}} \). Fig.3 and Fig.4 show the program and its output, respectively.
Figure 2 Numerical maximization of the log likelihood of a binomial model.

implicit real*8 (a-h,o-z)
parameter (ipmax=5)
dimension g(ipmax), vd(ipmax,ipmax+5)
dimension a1(ipmax),step(ipmax)
common /cmodel/ m,m1
external model0
m = 76
m1 = 12
do 1 i=1,ipmax
   step(i) = 0.001d0
   ip1=1
   a1(i) = 0.9d0
   call dall(model0,a1,ip1,el,g,vd,endc,step,0,3)
1 stop
end
subroutine model0(par,np,el,lel)
implicit real*8 (a-h,o-z)
common /cmodel/ m,m1
dimension par(np)
lel = 0
di = par(i)
if(di .le. 0.d0) return
if(di .ge. 1.d0) return
The following function is an artificial function. Not a real log likelihood function.

\[ f(x, y) = -\frac{1}{2}(r(x^2 + x + y)^2 + (x^2 + 3x + y)^2) \]  

Here, \( r = 40.0 \). The function takes the maximum value 0.0 at point B(0,0) shown in the top-left panel of Fig.5-1. The initial guess is taken at point A(−2.5, −3.0). Two contour curves in dotted line at 1.0 and 10.0 below the highest point are drawn in the figure.
Character ‘C’ in the panel titled NCOUNT = 51, indicates the position of the point $x_{51}$. The solid oval is the contour line at the level of $\phi_*$ $-$ 10.0 of the model (9) estimated based on information obtained up to this moment. The function value at point $x_*$, the center of the oval, is less than the value at $x_{51}$ and it turned out that this model is not good and $x_{52}$ is set equal to $x_{51}$. The model is modified to get one shown the in figure titled NCOUNT = 52, the function value at $x_*(D)$ is larger than that at $x_{52}$ and $x_{53}$ is set equal to $x_*$.

Iterating this process (see Figs. 5-2 and 5-3), we get the result shown in the figure titled NCOUNT=70 at top-right panel. Two ovals in solid line in the figure are contour lines at levels of $\phi_*$ $-$ 1.0 and $\phi_*$ $-$ 10.0. It is clear that fitting of the last model is poor globally in this case. But the local structure of the object function is captured by the model.
The print out of DALL is shown in Figure 3. The “graph” composed of * and X at the end of the figure shows profile of the object function along the line from point A to B. Since this “graph” does not show the function values at points $x_1, x_2, \ldots$, it does not increase monotonically. Seeing this “graph” we can get some information if the object function has an easy shape or not.
Figure 5-3. (continued)

<table>
<thead>
<tr>
<th>NCOUNT</th>
<th>PHI</th>
<th>SPHI-PHI</th>
<th>RHO</th>
<th>LAMBDA-1</th>
<th>ISPHI</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>-7.60</td>
<td>0.1857</td>
<td>-0.2604</td>
<td>-0.50</td>
<td>0</td>
</tr>
<tr>
<td>48</td>
<td>-7.41</td>
<td>0.6270</td>
<td>-0.0819</td>
<td>2.73</td>
<td>0</td>
</tr>
<tr>
<td>49</td>
<td>-6.79</td>
<td>0.2991</td>
<td>-0.2897</td>
<td>3.00</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>-6.49</td>
<td>1.1112</td>
<td>-0.0496</td>
<td>3.00</td>
<td>0</td>
</tr>
<tr>
<td>51</td>
<td>-5.38</td>
<td>-1.0834</td>
<td>-49.6538</td>
<td>-0.75</td>
<td>0</td>
</tr>
<tr>
<td>52</td>
<td>-2.36</td>
<td>1.8781</td>
<td>-1.8523</td>
<td>3.00</td>
<td>0</td>
</tr>
<tr>
<td>53</td>
<td>-2.69</td>
<td>-8.7070</td>
<td>-390.8669</td>
<td>-0.75</td>
<td>0</td>
</tr>
<tr>
<td>54</td>
<td>-4.01</td>
<td>1.0407</td>
<td>-0.5745</td>
<td>2.85</td>
<td>0</td>
</tr>
<tr>
<td>55</td>
<td>-2.97</td>
<td>-2.2384</td>
<td>-73.4456</td>
<td>-0.75</td>
<td>1</td>
</tr>
<tr>
<td>56</td>
<td>-2.97</td>
<td>0.7719</td>
<td>-0.4114</td>
<td>2.26</td>
<td>0</td>
</tr>
<tr>
<td>57</td>
<td>-2.19</td>
<td>2.5853</td>
<td>-4.6385</td>
<td>-0.75</td>
<td>0</td>
</tr>
<tr>
<td>58</td>
<td>-1.36</td>
<td>0.4698</td>
<td>-0.5283</td>
<td>3.00</td>
<td>0</td>
</tr>
<tr>
<td>59</td>
<td>-1.94</td>
<td>0.7766</td>
<td>-0.0683</td>
<td>-0.75</td>
<td>0</td>
</tr>
<tr>
<td>60</td>
<td>-1.12</td>
<td>0.2411</td>
<td>-0.1684</td>
<td>3.00</td>
<td>0</td>
</tr>
<tr>
<td>61</td>
<td>-1.66</td>
<td>0.4728</td>
<td>-0.2189</td>
<td>3.00</td>
<td>0</td>
</tr>
<tr>
<td>62</td>
<td>-0.65</td>
<td>0.0453</td>
<td>-3.1124</td>
<td>-0.75</td>
<td>0</td>
</tr>
<tr>
<td>63</td>
<td>-0.60</td>
<td>0.3932</td>
<td>-0.0956</td>
<td>-0.71</td>
<td>0</td>
</tr>
<tr>
<td>64</td>
<td>-0.18</td>
<td>0.0763</td>
<td>0.0233</td>
<td>3.00</td>
<td>0</td>
</tr>
<tr>
<td>65</td>
<td>-0.13</td>
<td>-0.0422</td>
<td>-2.91</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>66</td>
<td>-0.10</td>
<td>0.0882</td>
<td>0.0166</td>
<td>0.92</td>
<td>0</td>
</tr>
<tr>
<td>67</td>
<td>-0.02</td>
<td>0.0025</td>
<td>-0.0915</td>
<td>-0.75</td>
<td>0</td>
</tr>
<tr>
<td>68</td>
<td>-0.01</td>
<td>0.0123</td>
<td>-0.0015</td>
<td>3.00</td>
<td>0</td>
</tr>
<tr>
<td>69</td>
<td>0.00</td>
<td>-0.0001</td>
<td>-0.0065</td>
<td>-0.82</td>
<td>0</td>
</tr>
<tr>
<td>70</td>
<td>0.00</td>
<td>0.0014</td>
<td>0.0000</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

D2: Bill ended with abs(RHO) = 0.000003 < 0.00010 = rhomin: code = 0
3.3 Constrained Gaussian model fitting

Let \( x_1, \ldots, x_n \) be data. The log likelihood of a normal distribution model with respect to the data is given by

\[
    l_x(\mu_x, \sigma_x^2) = -\frac{n}{2} \log 2\pi - \frac{n}{2} \log \sigma_x^2 - \frac{1}{2\sigma_x^2} \sum_{i=1}^{n_x} (x_i - \mu_x)^2.
\]

(11)

For data \( y_1, \ldots, y_n \):

\[
    l_y(\mu_y, \sigma_y^2) = -\frac{n}{2} \log 2\pi - \frac{n}{2} \log \sigma_y^2 - \frac{1}{2\sigma_y^2} \sum_{i=1}^{n_y} (y_i - \mu_x)^2.
\]

(12)

MLE’s of \( \mu_x \) and \( \sigma_x^2 \) are

\[
    \hat{\mu}_x = \frac{1}{n} \sum_{i=1}^{n} x_i
\]

and

\[
    \hat{\sigma}_x^2 = \frac{1}{n_x} \sum_{i=1}^{n_x} (x_i - \hat{\mu}_x)^2,
\]

Similar expression for \( \mu_y, \sigma_y^2 \) can be given.

For data \( x = \{0.73, -0.06, 1.04, 2.29, 0.51, -0.45, 1.03, 0.44, 0.02, 0.11, -2.42\} \) MLE’s are

\[
    \hat{\mu}_x = 0.2945 \\
    \hat{\sigma}_x^2 = 1.2267
\]

and MLE’s for data \( y = \{0.10, 0.56, -1.11, -0.48, 3.46, -2.39, 0.36, 4.56\} \) are

\[
    \hat{\mu}_y = 0.6325 \\
    \hat{\sigma}_y^2 = 4.6491.
\]

The log likelihood for the model with the constraint \( \mu_x = \mu_y \) is given by

\[
    l_{x+y}(\mu_{x+y}, \sigma_x^2, \sigma_y^2) = -\frac{n}{2} \log 2\pi - \frac{n}{2} \log \sigma_x^2 - \frac{1}{2\sigma_x^2} \sum_{i=1}^{n_x} (x_i - \mu_{x+y})^2 \\
    -\frac{n}{2} \log 2\pi - \frac{n}{2} \log \sigma_y^2 - \frac{1}{2\sigma_y^2} \sum_{i=1}^{n_y} (y_i - \mu_{x+y})^2 \\
    = l_x(\mu_{x+y}, \sigma_x^2) + l_y(\mu_{x+y}, \sigma_y^2).
\]

(13)
The problem of analytical maximization of the function is equivalent to solving an algebraic equation of order three. Instead, let DALL solve the problem. Prepare subroutines shown in Fig. 7.

Subroutines llx, lly, and llxy are composed so that they compute log likelihood for parameter values specified by the array p of length m. As eq. (13) show the subroutine llxy is composed so that it calls llx and lly. The main routine and results are shown as Figures 8 and 9, respectively. From the resultant relation

\[ AIC_x + AIC_y = 37.46 + 39.00 = 76.46 > 74.62 = AIC_{xy} \]

supports the assumption \( \mu_x = \mu_y \).

```fortran
subroutine llx(p,m,el,lel)
    implicit real*8 (a-h,o-z)
    dimension p(m)
    common /i721/ n
    common /r721/ x(11)
    data pi /3.14159265d0/
    lel=0
    if(p(2) .gt. 0.d0) then
        sum=0.d0
        do 1 i=1,n
            sum=sum+(x(i)-p(1))**2
        1 continue
        el= -n*0.5*( dlog(pi*2)+dlog(p(2)) ) - 0.5*sum/p(2)
    lel=1
    end if
    return
end

subroutine lly(p,m,el,lel)
    implicit real*8 (a-h,o-z)
    dimension p(m)
    common /i722/ n
    common /r722/ x(8)
    data pi /3.14159265d0/
    lel=0
    if(p(2) .gt. 0.d0) then
        sum=0.d0
        do 1 i=1,n
            sum=sum+(x(i)-p(1))**2
        1 continue
        el= -n*0.5*( dlog(pi*2)+dlog(p(2)) ) - 0.5*sum/p(2)
    lel=1
    end if
    return
end

subroutine llxy(p,m,el,lel)
    implicit real*8 (a-h,o-z)
    dimension p(m),px(2),py(2)
    px(1)=p(1)
    px(2)=p(2)
    call llx(px,2,elx,lel)
    if(lel .eq. 0) return
    py(1)=p(1)
    py(2)=p(3)
    call lly(py,2,ely,lel)
    if(lel .eq. 0) return
    el=elx+ely
    return
end

block data ex72
    implicit real*8 (a-h,o-z)
    common /i721/ n1
    common /r721/ x1(11)
    common /i722/ n2
    common /r722/ x2(8)
    data n1/11/
    data x1/ 0.73, -0.06, 1.04, 2.29, 0.51,*
    * -0.45, 1.03, 0.44, 0.02, 0.11,*
    * -2.42 /
    data n2/8/
    data x2/ 0.10, 0.56, -1.11, -0.48, 3.46,*
    * -2.39, 0.36, 4.56 /
end

Figure 7. Subroutines to compute log likelihood of Gaussian models and constrained Gaussian model.

parameter (ipmax=4)
implicit real*8(a-h,o-z)
```

12
external llx,lly,llxy

myid = 0
do 1 i=1,ipmax
1 step(i) = 0.001d0
ipx=2
ax(1) = 0.d0
ax(2) = 1.d0
if(myid .eq. 0) then
  write(6,*) ' meanx(mle)=',ax(1)
  write(6,*) ' variancex(mle)=',ax(2)
  write(6,*) ' maximum likelihood=',el
  write(6,*) ' number of parameter=',ipx
  aic = -2*el + 2*ipx
  write(6,('' AICx='',F10.2)') aic
end if
ipy=2
ay(1) = 0.d0
ay(2) = 1.d0
if(myid .eq. 0) then
  write(6,*) ' meany(mle)=',ay(1)
  write(6,*) ' variancey(mle)=',ay(2)
  write(6,*) ' maximum likelihood=',el
  write(6,*) ' number of parameter=',ipy
  aic = -2*el + 2*ipy
  write(6,('' AICy='',F10.2)') aic
end if
ipxy=3
axy(1) = (ax(1) + ay(1))*0.5d0
axy(2) = ax(2)
axy(3) = ay(2)
if(myid .eq. 0) then
  write(6,*) ' meanxy(mle)=',axy(1)
  write(6,*) ' variancedx1(mle)=',axy(2)
  write(6,*) ' variancey(mle)=',axy(3)
  write(6,*) ' maximum likelihood=',el
  write(6,*) ' number of parameter=',ipxy
  aic = -2*el + 2*ipxy
  write(6,('' AICxy='',F10.2)') aic
end if
stop
end

Figure 8. A main program to fit Gaussian distribution model and constrained Gaussian distribution model.
meanx(mle) = 0.29454545454562
variancex(mle) = 1.2248781111300
maximum likelihood = -16.732202448716
number of parameter = 2
AICx = 37.46

Bill ended with abs(RHO) = 0.00006 < 0.00010 = rhomin: code = 0
Bill ended with abs(RHO) < rhomin.

meanx(mle) = 0.63250000002579
variancey(mle) = 4.6284090071286
maximum likelihood = -17.498215714685
number of parameter = 2
AICy = 39.00

Bill ended with abs(RHO) = 0.00000 < 0.00010 = rhomin: code = 0
Bill ended with abs(RHO) < rhomin.
4 Distributed files

DALL package summarized in Table 5 can be obtained at http://www.ism.ac.jp/software/ismlib/soft.e.html

<table>
<thead>
<tr>
<th>file name</th>
<th>contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>README</td>
<td>Latest announcement</td>
</tr>
<tr>
<td>example3.f</td>
<td>log likelihood subroutines and main program for the third example</td>
</tr>
<tr>
<td>gauss.c</td>
<td>and DALL routine in FORTRAN</td>
</tr>
<tr>
<td>dall_pack.c</td>
<td>log likelihood subroutines and main program for the third example in C</td>
</tr>
<tr>
<td>nrutil.h</td>
<td>a header file required by dall_pack.c</td>
</tr>
<tr>
<td>Makefile</td>
<td>sample makefile for C version for Unix environment</td>
</tr>
<tr>
<td>dall.ps</td>
<td>post script file of this monograph</td>
</tr>
</tbody>
</table>

Note: FORTRAN version DALL routine contained in this file is the one extracted from ARdock package. Those users who are interested in using DALL on a parallel computer are invited to down-load the source code of ARdock package.

ARdock, an Auto-Regressive model analyzer
Copyright_OML 1999 Makio Ishiguro, Hiroko Kato & Hirotugu Akaike
(http://www.ism.ac.jp/software/ismlib/soft.e.html)

So far, the test of DALL is successfully done in environments summarized in the following Table.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Operating System</th>
<th>Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>HITACHI SR8000</td>
<td>HI-UX/MPP</td>
<td>FORTRAN90</td>
</tr>
<tr>
<td>HP9000V2250</td>
<td>HP-UX 11.0</td>
<td>HP Fortran 90</td>
</tr>
<tr>
<td>IBM R/6000 SP</td>
<td>AIX Version 4</td>
<td>AIX XL Fortran</td>
</tr>
<tr>
<td>SGI Origin 2000/4-DS-I</td>
<td>IRIX Release 6.4 IP27</td>
<td>FORTRAN90</td>
</tr>
<tr>
<td>SPARCstation 5</td>
<td>Sun OS 4.1.4-JLE 1.1.4</td>
<td>g77 version0.5.23</td>
</tr>
<tr>
<td>PC-9801 nx/C</td>
<td>MS-DOS 5.00A-H</td>
<td>Microsoft FORTRAN Version 4.01</td>
</tr>
</tbody>
</table>

4.1 Licencing policy

We make public the source code of DALL under the open market licence policy proposed by the Institute of Statistical Mathematics.

Open Market Licence for software(version:OML-SW-E-1996)

0. In the following, “Copyright_OML notice” means “the copyright notice with a statement saying that the said software is made public under this Open Market Licence for software”.

1. On the condition that “Copyright_OML notice” and attached statements are copied as they are, all or any portion of the said software can be copied or redistributed.

2. Modification of the software is permitted, provided that the exact place of the modification, the date and the name of the modifier are shown conspicuously.

3. It is forbidden to apply for a patent on the software which utilizes the said software.
4. The said software is distributed without any warranty. The copyright holder takes no responsibility for any damage caused by the use of the said software.

5. Any interactively controlled application made utilizing the said software has to show “Copyright OML notice” and attached statements conspicuously when it is started up.

For the healthy development of software science, suitable form of the licencing is important. We definitely would like to claim our copyright to DALL. And we would like to use our copyright to let all concerned people use DALL source code freely under two conditions that (1) our credit is properly referred and (2) statements attached to the copyright note saying that the original DALL source code is distributed from the specified site without charge are kept as they are. We permit modification or redistribution of DALL source code as far as these two conditions are met. The redistribution may be with or without charge.

[Recommended form of reference/acknowledgment] When you are to publish software xxxxx, which utilizes DALL, here is an example of making reference to DALL.

xxxxx Copyright, 20yy uuuuu
xxxxx utilizes DALL, with here and there modifications. The places of modifications are indicated in the code.

DALL A subroutine for statistical model builders
Copyright_OML, 1996, 1999 M. Ishiguro and H. Akaike
The source code of this software can be obtained from ISMLIB(1) of the Institute of Statistical Mathematics without any charge. On the conditions that terms of the OPEN MARKET LICENCE(2) for software are observed and that the copyright holder nor the Institute of Statistical Mathematics take no responsibility on any result from the use, the program can be used or modified freely. On the condition that this note is attached as it is, DALL can be redistributed.

(1) http://www.ism.ac.jp/software/ismlib/soft.e.html

5 Acknowledgement

A part of this work is supported by the Grant-in-Aid for Scientific Research provided by The Ministry of Education, Science and Culture. Conversion of FORTRAN version DALL to C version was done by Mr. Yasumasa Matsuda of Kanagawa University. We would like to thank Ms. Syeda shahanra Hug of the Graduate University for Advanced Studies for her comments of the text.

References


