PROGRAMS FOR COMPUTING GROUP SEQUENTIAL BOUNDS USING THE LAN-DEMETS METHOD

David M. Reboussin
David L. DeMets
Kyungmann Kim
K.K. Gordon Lan

MADISON, WISCONSIN
Programs for Computing Group Sequential Bounds
Using the Lan–DeMets Method

by

David M. Reboussin
David L. DeMets
Department of Biostatistics
University of Wisconsin
Madison, WI 53706

Kyungmann Kim
Department of Biostatistics
Harvard School of Public Health
Cambridge, MA 02115

and

K. K. Gordon Lan
Department of Statistics / Computer and Information Systems
The George Washington University
Washington D.C. 20052

Biostatistics Technical Report No. 60

June 22, 1992
Abstract

Fortran 77 programs performing computations related to Lan–DeMets bounds are described. One program, **LANDEM**, produces bounds appropriate for user-supplied interim analysis times and type I error spending function. The other, **GLAN** computes probabilities associated with user-supplied interim analysis times and boundaries. Use of the programs is described, along with some of the theory on which they are based. Accuracy and validity of the results are assessed. Complete listings appear in the appendices.

**KEY WORDS:** Group sequential testing; Statistical computing; Type I error probability spending function; Use function.

1 Introduction

This report documents two programs designed to perform computations related to group sequential analysis of clinical trials. They are modifications of programs written by Dr. David DeMets and Dr. Kyungmann Kim. These new versions were designed 1) to update the code using Fortran77, 2) to increase the numerical efficiency, and 3) to create a “user-friendly” collection of subroutines which could be tailored to individual needs by interested researchers.

Repeated significance tests, as introduced by Armitage, McPherson & Rowe (1969), and the closely related group sequential methods described by Pocock (1977), O’Brien & Fleming (1979) and Lan & DeMets (1983), among others, all require some numerical integration for their application. Lan & DeMets (1983) provide a framework in which all these approaches may be considered, and the programs described here, originally written for that paper and for the paper by Kim & DeMets (1987b), essentially implement this framework in Fortran77.

The primary goal has been a program easy to use for routine analysis and possible to modify for specialized research. Users not interested in the Fortran code can concentrate on the explanations of the input and output, and on the transcripts of some typical interactive sessions. For more advanced work, the code in these programs contains a fair number of comment lines, but it will probably not be obvious. The details presented in the appendices are intended to provide assistance in gaining a deeper understanding of the programs and the theory underlying them.

The context in which these computations occur has been described simply and practically by Lan, Reboussin & DeMets (1992). Implicit in the methods proposed by Pocock (1977) and O’Brien & Fleming (1979) is the assumption that interim analyses are equally spaced in terms of the information available from the data. In order to relax this assumption and allow for more flexible and unequally spaced bounds, Lan & DeMets (1983) introduced type I error probability spending functions, denoted $\alpha^*(\tau)$. The argument $\tau$ represents the fraction of the total information available at each analysis. Denoting the information available for interim analyses $1, 2, \ldots, K$ as $i_1, i_2, \ldots, i_K$, and the total information by $I$, we can define the information fraction for analysis $k$ as $\tau_k = i_k / I$. Given $b_1, b_2, \ldots, b_{K-1}$, the Lan–DeMets
method finds $b_k$ such that
\[
\Pr(Z_1 \geq b_1 \text{ or } Z_2 \geq b_2 \text{ or } \ldots \text{ or } Z_k \geq b_k) = \alpha^*(i_k/I).
\]

If $\alpha^*(\tau)$ is the spending function, the type I error probability at the $k^{th}$ interim analysis should be $\alpha^*(i_k/I)$. In a “maximum information trial”—that is, a trial designed to end when a certain number of patients or number of deaths have been observed—$I$ is known. However, in a “maximum duration trial”—that is, a trial designed to end at a specific calendar time $T$—$I$ is not usually known. In either case the correlation structure for $Z_1, Z_2, \ldots, Z_k$, in equation (1) is observable, but if $I$ is not known, the percentage of elapsed calendar time for an analysis at calendar time $t_k$ is used instead:
\[
\alpha^*(\tau_k) \approx \alpha^*(\hat{t}_k) = \alpha^*(t_k/T).
\]

Section 2 describes sample input and output using the data from the Beta–Blocker Heart Attack Trial (BHAT). Section 3 contains sample input and output from the programs. Appendix A provides some background on the theory used to develop these methods. Appendix B describes the structure and function of each program. Appendix C presents testing of the programs for computational accuracy and simulations results for validity. Appendices D, E, and F contain listings of the code.

2 Using the Programs

We describe how to run the two programs using data from a sequentially monitored clinical trial. LANDEM generates boundaries sequentially as described by Lan & DeMets (1983) and Lan et al. (1992). GLAN evaluates the stopping probabilities for a given set of group sequential boundaries. As an example of the use of these programs, we will analyze data from the Beta–Blocker Heart Attack Trial or BHAT (Beta–Blocker Heart Attack Trial Research Group, 1982). BHAT, a study sponsored by the National Heart, Lung and Blood Institute, was designed to test whether long term use of propranolol by patients with recent heart attack reduced mortality. From June 1978 to October 1980, 3837 patients were randomized to either propranolol (1916 patients) or placebo (1921 patients). Follow-up was originally scheduled to end in June 1982. The total information D (number of deaths by June 1982) was never observed since the trial was terminated early in October 1981. The value of D was postulated to be 628 when BHAT was designed, but with the data available in September 1982, was estimated to be around 400 (Lan & DeMets, 1989). In the six Policy and Data Monitoring Board meetings (May 1979, October 1979, March 1980, October 1980, April 1981, and October 1981), the observed number of deaths were (56, 77, 126, 177, 247, 318) and normalized log-rank statistics were (1.68, 2.24, 2.37, 2.30, 2.34, 2.82).

2.1 Using LANDEM

The notation below resembles that in Lan & DeMets (1989). $t_c$ denotes calendar time measured from the beginning of the trial, and $T_c$ denotes the maximum duration in calendar
time. \( \tau \) is the information fraction or “information time”, which must often be estimated by some function either of calendar time or number of observed patients or events, denoted \( \hat{\tau} \). We shall first do an example using only calendar time.

**Example with calendar time**

We take \( t_c = 0 \) in June 1978 and assume the maximum duration is \( T_c = 48 \) months, which corresponds to June 1982. Then the calendar times for interim analyses correspond to (11, 16, 21, 28, 34, 40) months after the start of the trial. We estimate \( \tau \) as a function of calendar time by \( \hat{\tau} = t_c/T_c = t_c/48 \), so the information times are \( 0.2292, 0.3333, 0.4375, 0.5833, 0.7083, 0.8333 \), and adopt the use function \( \alpha^*(\tau) = \alpha \tau \) to construct a data monitoring boundary. This corresponds to \( \alpha^*_5(\tau) \) in Lan & DeMets (1983) and Kim & DeMets (19876). The original BIHAT design had a two-sided significance level of 0.05.

When the data were monitored in May 1979, \( t_{c1} = 11, \hat{\tau}_1 = 11/48 = 0.2292 \) and \( \alpha^*(\hat{\tau}_1) = 0.025 \times 0.2292 = 0.0057 \). This leads to a boundary value of \( c_1 = 2.53 \) from the normal table: if \( Z_1 \) is standard normal, \( P(Z_1 \geq 2.53) = 0.0057 \). The program **LANDEP** calculates \( c_1 \).

In October 1979, \( t_{c2} = 16, \hat{\tau}_2 = 16/48 = 0.3333, \) and \( \alpha^*(\hat{\tau}_2) = 0.0083 \). Ignoring the observed number of deaths and using only calendar time, the calculation proceeds as follows. Suppose \( Z_1 \) and \( Z_2 \) are standard normal with correlation coefficient \( \rho_{12} = \sqrt{0.2292/0.3333} = 0.8293. \) We wish to find \( c_2 \) such that

\[
P(Z_1 \geq 2.53 \text{ or } Z_2 \geq c_2) = 0.0083
\]

This solution requires some numerical integration which the program performs. In fact, this equality is satisfied if \( c_2 = 2.61 \). For this example, **LANDEP** would ask for

- the overall significance level (.05)
- whether the test was one-sided or two-sided symmetric (2),
- which \( \alpha^* \) function to apply (\( \alpha^*_5 \)),
- how many analyses were done in the past (1),
- what the previous analysis times were (.2292),
- what the current analysis time is (.3333).

- whether exact or estimated information will be entered (Lan and DeMets, 1989) (no),

So, at the second interim analysis, the program returns \( c_1 = 2.53 \) and \( c_2 = 2.61. \) At the third, we enter 0.2292 and 0.3333 as previous times, and 0.4375 as the current time. The program returns \( c_1 \) and \( c_2 \) as before, plus \( c_3 = 2.57. \) Continuing in this manner, we obtain the boundary values \( 2.53, 2.61, 2.57, 2.47, 2.43, 2.38 \).
Example with information

We now repeat the above calculation using the information in the number of deaths. Assuming \( D = 628 \), the information fractions are \( (56/628, 77/628, 126/628, 177/628, 247/628, 318/628) \), or \( (0.0892, 0.1226, 0.2006, 0.2818, 0.3933, 0.5064) \). Then at the second interim analysis, the program would ask for

- the overall significance level (.05)
- whether the test was one-sided or two-sided symmetric (2),
- what \( \alpha^* \) function to apply (\( \alpha^*_2 \)),
- how many analyses were done in the past (1),
- what the previous analysis times were (.0892),
- what the current analysis time is (.1226).

whether exact or estimated information will be entered (Lan and DeMets 1989) (no).

The information fractions are treated as times. Since we do not enter the information separately apart from the information fractions, the answer to the last question is “no”. The output boundary values are (2.84, 2.97). At the sixth analysis, when the additional times are input, the resulting boundary values are (2.84, 2.97, 2.79, 2.72, 2.61, 2.54).

Some users may be familiar with the use of both information and calendar time as described in Lan & DeMets (1989) and Lan et al. (1992). The program includes such an option. We will use the percent of elapsed calendar time to determine how much type I error probability is to be spent, but for the correlation of successive test statistics, we will use the information in the number of deaths. The first bound is computed exactly as above. For the analysis in October 1979, \( t_{c2} = 16, \bar{r}_2 = 16/4 = 0.3333, \) and \( \alpha^*(\bar{r}_2) = 0.0083 \) also just as before. To evaluate \( c_2 \), note that even though \( r_2 = 77/D \) is unknown, \( \tau_1/\tau_2 = 56/77 = 0.7273 \) is observed. If \( Z_1 \) and \( Z_2 \) are standard normal with correlation coefficient \( \rho_{12} = \sqrt{0.7273} = 0.8528 \), the solution to

\[
P(Z_1 \geq 2.53 \text{ or } Z_2 \geq c_2) = 0.0083
\]

is \( c_2 = 2.59 \). The program asks exactly the same questions as before. Since the times entered were based on the percent of elapsed calendar time, it is desirable to use the information available in the number of deaths. When the question on exact or estimated information is asked, we answer “yes” and enter

- what the information at previous analyses was, and
- what the current information is.

The information is the number of deaths in this example. The resulting bounds are (2.53, 2.59, 2.63, 2.50, 2.51, 2.47) for the six data monitoring points of BHAT, and this boundary is crossed at \( t_{c6} = 40 \) or in October of 1981. This is exactly the result given for the example in Lan & DeMets (1989).
2.2 Using GLAN

Users may sometimes wish to evaluate the stopping and continuation probabilities associated with a given boundary, possibly under various assumptions about the underlying mean of the observed process. Given boundary values \(c_1, c_2, \ldots, c_k\), suppose we wish to find \(p_1, \ldots, p_n\) for analysis times \(t_1, \ldots, t_n\). The problem amounts to computing the quantity

\[
P(Z_1 \geq c_1 \text{ or } \ldots \text{ or } Z_k \geq c_k) = p_k
\]

for normal variates \(Z_1, \ldots, Z_n\) having mean \(\Delta t_k\) and \(\text{Cov}(Z_j, Z_k) = \min(t_j, t_k)\).

For example, suppose we are given the boundaries output from the first example above, \((2.53, 2.61, 2.57, 2.47, 2.43, 2.38)\) at analysis times \((0.2292, 0.3333, 0.4375, 0.5833, 0.7083, 0.8333)\) and we wish to know the stopping probabilities under the hypothesis \(\Delta = 2.5\). The program GLAN will ask for

- the number of analyses (6),
- the information times of the analyses (.2292, .3333, etc.),
- whether a nonzero drift parameter is to be used (yes),
- the value of the drift parameter (2.5),
- whether the boundary is one or two sided (2),
- whether the two sided boundary is symmetric (yes).
- whether the bounds are to be input from a file (no).
- the bounds to be evaluated (2.53, 2.61, etc.).

The resulting probabilities are \((0.0915, 0.1507, 0.2225, 0.3358, 0.4311, 0.5221)\).

Once probabilities associated with the bounds and drift parameter have been output, the program asks whether the user wishes to recompute using a new drift parameter. This is a useful feature when confidence intervals are desired, since this requires varying the drift parameter until the accumulated probability of exceeding the final value attains the desired alpha level (Kim & DeMets, 1987a).

Verbatim transcripts of interactive sessions with the two programs are included in the next section.

3 Examples of running the programs

This appendix contains four examples of interactive sessions with the programs: two with LANDEM and two with GLAN. In each case, the data used was taken from Lan & DeMets (1989).
3.1 Calendar time in LANDEM

This is an interactive session with LANDEM using the BIIAT data and calendar time as the only time scale. The input sequence is described in Section 2.

Is this an interactive session? (1=yes,0=no)
1
interactive = 1
Overall significance level? (>0 and <=1)
.05
alpha = 0.050
One-sided or two-sided symmetric?
2
2-sided test
Use function? (1-5)
(1) Obrien-Fleming type
(2) Pocock type
(3) alpha * t
(4) alpha * t^-1.5
(5) alpha * t^-2
3
Use function alpha-star
Number of interim analyses in the past (0 if this is the first):

5 previous analyses.
Enter times (or information fractions > 0 and < 1) in the past:
.2292 .3333 .4375 .5667 .7063
Previous times (or info fractions) are 0.229 0.333 0.438 0.583 0.708
Time or info fraction (< 1) of the current interim analysis?
5233
Current time (or information fraction) is 0.93
Do you wish to input the exact or estimated information? (e.g. number of patients or number of events, as in Lan & DeMets 897)
(1=yes, 0=no)

Delta will be taken to be zero.

This program generates two-sided symmetric boundaries.
n = 6
alpha = 0.050
use function for the lower boundary = 3
use function for the upper boundary = 3

<table>
<thead>
<tr>
<th>Time</th>
<th>Bounds</th>
<th>alpha(i)-alpha(i-1)</th>
<th>cum alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.23</td>
<td>-2.5294 2.5284</td>
<td>0.01146</td>
<td>0.01146</td>
</tr>
<tr>
<td>0.32</td>
<td>-2.6098 2.6098</td>
<td>0.00520</td>
<td>0.01666</td>
</tr>
<tr>
<td>0.44</td>
<td>-2.5689 2.5689</td>
<td>0.00521</td>
<td>0.02188</td>
</tr>
<tr>
<td>0.58</td>
<td>-2.4678 2.4678</td>
<td>0.00729</td>
<td>0.02917</td>
</tr>
<tr>
<td>0.71</td>
<td>-2.4298 2.4298</td>
<td>0.00625</td>
<td>0.03542</td>
</tr>
<tr>
<td>0.83</td>
<td>-2.3841 2.3841</td>
<td>0.00625</td>
<td>0.04167</td>
</tr>
</tbody>
</table>

In this case, the program outputs the number of analyses, the type I error specified, the use function chosen, the times, the computed boundaries, and the type I error "spent" at each analysis.

Some users may want to use the program noninteractively. This can be done by preparing an input file with the appropriate format. Each question is answered on its own line in the input file, and the answer to the first question must be "no" or "0". Here is an input file which reproduces the above interactive session:
The resulting output is

Is this an interactive session? (1=yes, 0=no)
interactive = 0
alpha = 0.050
2-sided test
Use function alpha-star 3
5 previous analyses.
Previous times (or info fractions) are 0.229 0.333 0.4375 0.5833 0.7083
Current time (or information fraction) is 0.8333
Delta will be taken to be zero.

This program generates two-sided symmetric boundaries.

\[ n = 6 \]

alpha = 0.050
use function for the lower boundary = 3
use function for the upper boundary = 3

<table>
<thead>
<tr>
<th>Time</th>
<th>Bounds</th>
<th>alpha(i)-alpha(i-1)</th>
<th>cum alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.23</td>
<td>-2.5284 2.5284</td>
<td>0.01146</td>
<td>0.01146</td>
</tr>
<tr>
<td>0.33</td>
<td>-2.6098 2.6098</td>
<td>0.00520</td>
<td>0.01666</td>
</tr>
<tr>
<td>0.44</td>
<td>-2.5689 2.5689</td>
<td>0.00521</td>
<td>0.02188</td>
</tr>
<tr>
<td>0.68</td>
<td>-2.4678 2.4678</td>
<td>0.00729</td>
<td>0.02917</td>
</tr>
<tr>
<td>0.71</td>
<td>-2.4298 2.4298</td>
<td>0.00625</td>
<td>0.03542</td>
</tr>
<tr>
<td>0.83</td>
<td>-2.3841 2.3841</td>
<td>0.00625</td>
<td>0.04167</td>
</tr>
</tbody>
</table>

3.2 Using information in LANDEm

For this session, the numbers of events were entered as information.

Is this an interactive session? (1=yes, 0=no)
interactive = 1
Overall significance level? (>0 and <1)
0.05
alpha = 0.050
One(1)– or two(2)-sided symmetric?
2
2-sided test
Use function? (1-5)
(1) O'Brien-Fleming type
(2) Pocock type
(3) alpha * t
(4) alpha * t^-1.5
(5) alpha * t^-2
2
Use function alpha-star 2
Number of interim analyses in the past (0 if this is the first):
5 previous analyses.
Enter times (or information fractions > 0 and < 1) in the past:
.2292 .3333 .4376 .5833 .7083
Previous times (or info fractions) are 0.229 0.333 0.438 0.583 0.708
Time or info fraction (<= 1) of the current interim analysis?
.8333
Current time (or information fraction) is 0.833
Do you wish to input the exact or estimated information? (e.g.
number of patients or number of events, as in Lan & DeMete 82?)
(y=yes, o=no)
1
Entering information.
Information for past analyses:
56 77 126 177 247
Previous information 56.000 77.000 126.000 177.000 247.000
Information for current analysis:
318
Current information 318.000

Delta will be taken to be zero.

This program generates two-sided symmetric boundaries.
n = 6
alpha = 0.050
use function for the lower boundary = 2
use function for the upper boundary = 2

<table>
<thead>
<tr>
<th>Time</th>
<th>Information</th>
<th>Bounds</th>
<th>alpha(i)-alpha(i-1)</th>
<th>cum alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.23</td>
<td>56.00</td>
<td>-2.3964</td>
<td>2.3964</td>
<td>0.01660</td>
</tr>
<tr>
<td>0.33</td>
<td>77.00</td>
<td>-2.5072</td>
<td>2.5072</td>
<td>0.00604</td>
</tr>
<tr>
<td>0.44</td>
<td>126.00</td>
<td>-2.5963</td>
<td>2.5963</td>
<td>0.00539</td>
</tr>
<tr>
<td>0.58</td>
<td>177.00</td>
<td>-2.5097</td>
<td>2.5097</td>
<td>0.00668</td>
</tr>
<tr>
<td>0.71</td>
<td>247.00</td>
<td>-2.5551</td>
<td>2.5551</td>
<td>0.00509</td>
</tr>
<tr>
<td>0.83</td>
<td>318.00</td>
<td>-2.5491</td>
<td>2.5491</td>
<td>0.00462</td>
</tr>
</tbody>
</table>

In addition to the output described previously, the information is also reported.

3.3 Computations from GLAN under an alternative

This interactive session with GLAN takes as input the output from the previous LANDEM run, and computes probabilities under the assumption that the underlying process has a drift parameter of 2.5.

Is this an interactive session? (y=yes, o=no)
1
interactive = 1
Number of interim analyses?
6
6 interim analyses.
Times of interim analyses:
.2292 .3333 .4376 .5833 .7083 .8333
Analysis times: 0.229 0.333 0.438 0.583 0.708 0.833
Do you wish to use drift parameter (Delta) other than zero? (y=yes, o=no)
1
Enter noncentrality parameter:
2.5
Delta = 2.500000000000000
One(t)= or two(2)-sided?
The number of analyses and the noncentrality parameter are output, along with a table of probabilities. The analysis number, time, and bounds from input are followed by columns containing, respectively, $Q_i + R_i$ and $\sum_{j=1}^{i} Q_j + R_j$ (see Appendix A).

### 3.4 Computations from GLAN under the null hypothesis

This GLAN run takes the output from the second LANDEM example as input, assuming the null hypothesis is true.
In this example, numbers of events were input instead of times on a (0,1] scale, but this causes no difficulty for the program, which simply divides each time by the last (and largest) time entered.

### 3.5 Using GLAN to find a confidence interval

When the null hypothesis is rejected, the test can be inverted to find a confidence parameter for the drift parameter. Details for the theory underlying this observation can be found in Kim & DeMets (1987a). The program GLAN provides for this by prompting for a new drift parameter after probabilities have been computed. By increasing and decreasing delta until the desired alpha level is attained, a confidence interval can be determined.

In the example above, using calendar time, the bounds computed by GLAN are exceeded at the sixth analysis. We replace the bound at the sixth analysis by the observed log-rank statistic, which was 2.82. We start with delta equal to zero and look for a 90 percent confidence interval.

Is this an interactive session? (1=yes, 0=no)

```
Is this an interactive session? (1=yes, 0=no)
1
interactive = 1
```

Number of interim analyses?

```
Number of interim analyses?
6
```

Times of interim analyses:

```
Times of interim analyses:
.2292 .3333 .4375 .5625 .7083 .8333
```

Analysis times:

```
Analysis times:
.2292  .3333  .4375  .5625  .7083  .8333
```

Do you wish to use drift parameter (Delta) other than zero? (1=yes, 0=no)

```
Do you wish to use drift parameter (Delta) other than zero? (1=yes, 0=no)
0
```

Delta = 0.000000000000000E+00

One(1)- or two(2)-sided?

```
One(1)- or two(2)-sided?
2
```

2-sided test

```
2-sided test
```

Symmetric bounds? (1=yes, 0=no)

```
Symmetric bounds? (1=yes, 0=no)
1
```

Two sided symmetric bounds.

Enter upper bounds in standardized form:
Bounds entered.

\[ n = 6, \delta = 0.0000 \]

<table>
<thead>
<tr>
<th>look</th>
<th>time</th>
<th>lower</th>
<th>upper</th>
<th>alpha(i) - alpha(i-1)</th>
<th>cum alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.23</td>
<td>-2.6288</td>
<td>2.6288</td>
<td>0.01145</td>
<td>0.01145</td>
</tr>
<tr>
<td>2</td>
<td>0.33</td>
<td>-2.6099</td>
<td>2.6099</td>
<td>0.00521</td>
<td>0.01666</td>
</tr>
<tr>
<td>3</td>
<td>0.44</td>
<td>-2.5690</td>
<td>2.5690</td>
<td>0.00521</td>
<td>0.02186</td>
</tr>
<tr>
<td>4</td>
<td>0.55</td>
<td>-2.4678</td>
<td>2.4678</td>
<td>0.00729</td>
<td>0.02915</td>
</tr>
<tr>
<td>5</td>
<td>0.71</td>
<td>-2.4298</td>
<td>2.4298</td>
<td>0.00625</td>
<td>0.03540</td>
</tr>
<tr>
<td>6</td>
<td>0.83</td>
<td>-2.8200</td>
<td>2.8200</td>
<td>0.00061</td>
<td>0.03601</td>
</tr>
</tbody>
</table>

Do you wish to recompute using a new drift parameter (\(\delta\)) (1=yes, 0=no)?

Enter new drift parameter:

0.5

Recomputing with \(\delta = 0.50000000000000000000\)

\[ n = 6, \delta = 0.5000 \]

<table>
<thead>
<tr>
<th>look</th>
<th>time</th>
<th>lower</th>
<th>upper</th>
<th>alpha(i) - alpha(i-1)</th>
<th>cum alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.23</td>
<td>-2.6288</td>
<td>2.6288</td>
<td>0.01365</td>
<td>0.01365</td>
</tr>
<tr>
<td>2</td>
<td>0.33</td>
<td>-2.6099</td>
<td>2.6099</td>
<td>0.00679</td>
<td>0.02043</td>
</tr>
<tr>
<td>3</td>
<td>0.44</td>
<td>-2.5690</td>
<td>2.5690</td>
<td>0.00720</td>
<td>0.02764</td>
</tr>
<tr>
<td>4</td>
<td>0.55</td>
<td>-2.4678</td>
<td>2.4678</td>
<td>0.01076</td>
<td>0.03849</td>
</tr>
<tr>
<td>5</td>
<td>0.71</td>
<td>-2.4298</td>
<td>2.4298</td>
<td>0.00964</td>
<td>0.04823</td>
</tr>
<tr>
<td>6</td>
<td>0.83</td>
<td>-2.8200</td>
<td>2.8200</td>
<td>0.00113</td>
<td>0.04936</td>
</tr>
</tbody>
</table>

Do you wish to recompute using a new drift parameter (\(\delta\)) (1=yes, 0=no)?

Enter new drift parameter:

4.5

Recomputing with \(\delta = 4.50000000000000000000\)

\[ n = 6, \delta = 4.5000 \]

<table>
<thead>
<tr>
<th>look</th>
<th>time</th>
<th>lower</th>
<th>upper</th>
<th>alpha(i) - alpha(i-1)</th>
<th>cum alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.23</td>
<td>-2.6288</td>
<td>2.6288</td>
<td>0.35404</td>
<td>0.35404</td>
</tr>
<tr>
<td>2</td>
<td>0.33</td>
<td>-2.6099</td>
<td>2.6099</td>
<td>0.17942</td>
<td>0.53346</td>
</tr>
<tr>
<td>3</td>
<td>0.44</td>
<td>-2.5690</td>
<td>2.5690</td>
<td>0.16210</td>
<td>0.69556</td>
</tr>
<tr>
<td>4</td>
<td>0.55</td>
<td>-2.4678</td>
<td>2.4678</td>
<td>0.15913</td>
<td>0.85469</td>
</tr>
<tr>
<td>5</td>
<td>0.71</td>
<td>-2.4298</td>
<td>2.4298</td>
<td>0.07286</td>
<td>0.92766</td>
</tr>
<tr>
<td>6</td>
<td>0.83</td>
<td>-2.8200</td>
<td>2.8200</td>
<td>0.01666</td>
<td>0.94321</td>
</tr>
</tbody>
</table>

Do you wish to recompute using a new drift parameter (\(\delta\)) (1=yes, 0=no)?

Enter new drift parameter:

4.6

Recomputing with \(\delta = 4.60000000000000000000\)

11
\[ n = 6, \ \text{delta} = 4.6000 \]

<table>
<thead>
<tr>
<th>look</th>
<th>time</th>
<th>lower</th>
<th>upper</th>
<th>alpha(i)-alpha(i-1)</th>
<th>cum alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.23</td>
<td>-2.5288</td>
<td>2.5288</td>
<td>0.37200</td>
<td>0.37200</td>
</tr>
<tr>
<td>2</td>
<td>0.33</td>
<td>-2.6099</td>
<td>2.6099</td>
<td>0.18379</td>
<td>0.55579</td>
</tr>
<tr>
<td>3</td>
<td>0.44</td>
<td>-2.5680</td>
<td>2.5680</td>
<td>0.16195</td>
<td>0.71774</td>
</tr>
<tr>
<td>4</td>
<td>0.58</td>
<td>-2.4678</td>
<td>2.4678</td>
<td>0.15335</td>
<td>0.87110</td>
</tr>
<tr>
<td>5</td>
<td>0.71</td>
<td>-2.4286</td>
<td>2.4286</td>
<td>0.06724</td>
<td>0.93834</td>
</tr>
<tr>
<td>6</td>
<td>0.83</td>
<td>-2.3200</td>
<td>2.3200</td>
<td>0.01411</td>
<td>0.95245</td>
</tr>
</tbody>
</table>

Do you wish to recompute using a new drift parameter (delta) (yes, own)?

0

Done.

Goodbye.

For delta = 0.5, the probability of rejecting is about 0.05, and for delta = 4.6 the probability of rejecting is about 0.95, so a 90 percent confidence interval for delta is (0.5, 4.6)

References


A Theory related to the computations.

Consider a process in continuous time, \( W(t), t \in [0, 1] \), having unknown drift parameter \( \Delta \), which may be inspected at times \( t_i, i = 1, 2, \ldots, n \). We wish to test the hypothesis \( H_0 : \Delta = 0 \) against some alternative sequentially; that is, performing the test at each inspection time \( t_i \) and proceeding only if the test fails to reject. For the case where \( W(t) \) is a Brownian motion this is the problem considered by Lan & DeMets (1983), by Kim & DeMets (1987b), and by Lan & DeMets (1989). \( W(t) \) will be considered a Brownian motion throughout.

We are interested in whether \( W(t) \) falls in some region of the real line, for example in \([a_i, b_i]\). Let \( A_1 \) and \( A_2 \) denote events, specifically, let \( A_i = \{ W(t_i) \in [a_i, b_i] \} \), so that \( A_i \) is the event that the process takes a value in \([a_i, b_i]\) at \( t_i \). We may think of \( a_i \) and \( b_i \) as sequential bounds, so that the sequential test rejects if \( W(t_i) \notin [a_i, b_i] \), i.e. \( \overline{A_i} \). Recall the definition of simple conditional probability:

\[
P(A_1 \cap A_2) = P(A_2 \mid A_1)P(A_1).
\]

This provides the basis for computing Lan–DeMets bounds. The left hand side is the probability of continuing past the second interim analysis; that is, the probability of being in the first acceptance region and the second acceptance region. The right hand side has two terms, the conditional probability of continuing past the second analysis given continuance past the first, multiplied by the probability of continuing past the first. Of course, this may be easily extended past the second analysis.

In order to be clear about the probabilities calculated, consider two equalities:

\[
1 = P(\overline{A_1}) + P(A_1 \cap \overline{A_2}) + P(A_1 \cap A_2)
\]

\[
= P(\overline{A_1}) + P(\overline{A_2} \mid A_1)P(A_1) + P(A_2 \mid A_1)P(A_1)
\]

(2)

and

\[
1 = P(\overline{A_2} \mid A_1) + P(A_2 \mid A_1)
\]

(3)

In each case, the probabilities sum to 1, and in the second, this summation corresponds to integrating a conditional density over its entire range. But it is the first of which we avail ourselves for the computations described below.

McPherson & Armitage (1971) denote the quantities above as follows. The first two terms in equation 2 are stopping probabilities, identified in the appendix of McPherson & Armitage (1971) as \( Q_1 + R_1 \) and \( Q_2 + R_2 \), where \( R_i \) is the probability of the event \( W(t_i) \in [-\infty, a_i] \) and...
$Q_i$ is the probability of the event $W(t_i) \in [b_i, \infty]$. The third is a continuation probability, which they denote $1 - P_2$. For more than two analyses, subscripts indicate the analysis number. Notice that $\sum_{j=1}^{i} Q_j + R_j = P_i$, and $P_i - P_{i-1} = Q_i + R_i$.

Lan & DeMets (1983) refer to Armitage et al. (1969), for the distribution theory. The main generalizations incorporated in LANDEM and GLAN are that inspection times need not be equally spaced (Lan & DeMets, 1983) and bounds may be symmetric, asymmetric or one-sided (Kim & DeMets, 1987b). Consider a sequence of bounds, $(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)$ to be applied at times $t_1, t_2, \ldots, t_n$. Let $g$ denote the standard normal density function,

$$g(u) = \frac{1}{\sqrt{2\pi}} \exp \left(-\frac{1}{2}u^2\right)$$

The function used to define $P_i$, $Q_i$ and $R_i$ is defined recursively as

$$f_1(x) = g(x)$$

$$f_i(x) = \int_{a_{i-1}}^{b_{i-1}} g((x - u)/\sigma_i) \sigma_i^{-1} f_{i-1}(u) \, du$$

(1)

where $\sigma_i$ is the square root of the variance of the process increment from $t_{i-1}$ to $t_i$, which is $\sqrt{t_i - t_{i-1}}$ since $\text{var}[W(t_i) - W(t_{i-1})] = t_i - t_{i-1}$. Then

$$1 - P_i = \int_{a_i}^{b_i} f_i(x) \, dx,$$

$$Q_i = \int_{b_i}^{\infty} f_i(x) \, dx,$$

and

$$R_i = \int_{-\infty}^{a_i} f_i(x) \, dx.$$

Both Armitage et al. (1969) and McPherson & Armitage (1971) refer to $f_i$ as a density function, but it is more properly a joint density function, describing the probability of values of $W(t_i)$ and the probability of stopping at $i$. In fact, integrating $f_i$ from $-\infty$ to $+\infty$ gives $(1 - P_i) + Q_i + R_i = 1 - P_{i-1}$, the probability of not stopping at or before the $(i-1)^{th}$ analysis. This point is made for the careful reader of the appendix in McPherson & Armitage (1971).

Notice that the covariance of the process at times $t_1, \ldots, t_n$ has only $n$ parameters, and that the process can be rescaled. In fact the covariance of $(W(t_1), \ldots, W(t_k))$ is

$$\begin{bmatrix}
    t_1 & t_1 & \ldots & t_1 \\
    t_1 & t_2 & \ldots & t_2 \\
    \vdots & \vdots & \ddots & \vdots \\
    t_1 & t_2 & \ldots & t_k
\end{bmatrix},$$

14
Dividing through by \( t_k \) gives

\[
\begin{bmatrix}
  t_1/t_k & t_1/t_k & \ldots & t_1/t_k \\
  t_2/t_k & t_2/t_k & \ldots & t_2/t_k \\
  \vdots & \vdots & \ddots & \vdots \\
  t_k/t_k & t_k/t_k & \ldots & 1
\end{bmatrix}.
\]

It is computationally convenient, as well as consistent with Lau and DeMets (1989) to make some rescalings of this sort.

Computations at the first analysis involve only the standard normal density and distribution function, but for the second and beyond, numerical integration is necessary. Previous versions of these programs have performed numerical integration when computing \( f_i \) and when computing \( P_i \). But by combining the two preceding equations and applying Fubini’s theorem, we have

\[
1 - P_i = \int_{a_i}^{b_i} \int_{a_i-1}^{b_i-1} g((x - u)/\sigma_i) \sigma_i^{-1} f_{i-1}(u)du
dx
= \int_{a_i}^{b_i} \int_{a_i}^{b_i} g((x - u)/\sigma_i) \sigma_i^{-1} f_{i-1}(u)du
dx
= \int_{a_i}^{b_i} [\Phi((b_i - u)/\sigma_i) - \Phi((a_i - u)/\sigma_i)] f_{i-1}(u)du
\]

(5)

This manipulation allows the use of simple, accurate approximations to the normal distribution function to be used for computing \( P_i \). Armitage et al. (1969) suggest a method related to this as an alternative to theirs.

It may be worth mentioning a technical but sticky point to conclude this discussion. When the programs ask for “standardized bounds”, they actually mean standardized under the null hypothesis. We illustrate using the upper bound only. In the program, the variable \( zb \) contains the “standardized” upper bounds, while the variable \( yb \) contains the upper limits of integration, \( b_i \) in the above. This is only a confusing issue if the drift parameter is different from zero, and so only arises in \textbf{GLAN}, since such an option was not implemented in \textbf{LANDEM}.

For the first analysis, which uses only the cumulative normal distribution, we have \( W(t_1) \sim N(\Delta \ast t_1, \sigma_1) \) with \( \sigma_1 = t_1 \). The probability calculated for exceeding the first boundary is

\[
P\left( \frac{W(t_1)}{\sigma_1} > zb(1) \right) = P(W(t_1) > zb(1) \ast \sigma_1)
= P\left( \frac{W(t_1) - \Delta \ast t_1}{\sigma_1} > \frac{zb(1) \ast \sigma_1 - \Delta \ast t_1}{\sigma_1} \right)
= P(Z > zb(1) - \Delta \ast t_1/\sigma_1)
\]

Meanwhile, to convert from \( zb \) to \( yb \), we use the formula

\[
yb(i) = zb(i) \ast \sigma_i - \Delta \ast t_i
\]

to convert between limits of integration and “standardized bounds”.

15
\section*{B Description of computations.}

In the programs, the sequence of computations given \((a_i, b_i), i > 1\), is roughly as follows. \(\sigma_i\) is computed. Then two completely independent subroutines are called: one to compute \(P_i\) and, if there are more analyses to come, one to compute \(f_i\).

For the routine computing \(P_i\), a grid of values of \(f_{i-1}(u)\) for \(u \in (a_{i-1}, b_{i-1})\), saved from the previous step, is needed. Notice that the grid size is standardized, so that it is finer when the process has a smaller standard deviation. At each grid point \(u\), the quantity

\[
[\Phi((b_i - u)/\sigma_i) - \Phi((a_i - u)/\sigma_i)] f_{i-1}(u)
\]

is computed and stored in an array. This array is then passed to a numerical integration routine along with \(a_{i-1}, b_{i-1}\) and the grid size, and \(1 - P_i\) in Equation 5 is returned.

\(f_i\) is computed for a grid of values between \(a_i\) and \(b_i\). For each grid point, the grid of values of \(f_{i-1}\) is needed. Letting \(u\) denote a point in the grid from \(a_{i-1}\) to \(b_{i-1}\), and \(x\) denote a point in the grid from \(a_i\) to \(b_i\), the quantity

\[
f_{i-1}(u) g((u - x)/\sigma_i)/\sigma_i
\]

is computed and stored in an array. As before, this array is passed to a numerical integration routine, along with \(a_{i-1}, b_{i-1}\), and the grid size, and \(f_i(x)\) in Equation 4 is obtained and stored for the next step.

Currently, the numerical integration routine is a trapezoidal rule, which appears to produce fairly accurate results (see Appendix C), but users may substitute their own routines. Notice that routines which require different types of grids than that currently programmed will necessitate careful modification of several subroutines.

In Armitage et al. (1969), computations are described for equally spaced analyses and constant bounds. A simple and popular generalization of these computations is to allow bounds corresponding to those of O'Brien & Fleming (1979). These types of bounds can be evaluated using GLAN and inputting equally spaced times for the interim analyses. But GLAN also allows interim analyses to have any spacing. The drift parameter, which can be input to GLAN, is the mean of the process when \(t = 1\), so that at \(t_i\) the process has mean \(\Delta t_i\).

Although the essential computations are the same, LANDEM is a quite different program. Suppose a clinical trial is underway, and Lan–DeMets bounds are needed for an upcoming interim analysis. The user must input times of any previous interim analyses, time of the current analysis, and the Type I error spending rate function. The program then reports what bounds should be used to determine whether or not to stop the trial. This is accomplished by a trial and error search: making an initial choice of bounds, computing probabilities, altering bounds. As a result it is more time consuming. Users familiar with Lan & DeMets (1989) can specify percentage of elapsed calendar time to determine how much type I error is to be spent and information to specify the correlation of successive test statistics (see Section 2).
C  Program testing and some simulation results.

The programs were tested for both internal consistency and validity. Internal accuracy was verified by computing certain quantities redundantly and comparing them. Validity was verified by comparing computed probabilities with those generated by simulation. In addition, results from LANDEM were given as arguments to GLAN, and output from GLAN was compared to values reported by Armitage et al. (1969).

Some of the programs were slightly upgraded after these tests were performed, so users may not be able to reproduce the following result exactly. However, the updated programs are more accurate than the versions tested.

Internally, GLAN had good consistency. The number reported in the simulations below is numerically integrated “tails of the distribution” at each analysis; that is, \( Q_i + R_i \), the stopping probabilities defined in Appendix A. A comparison with \( P_i - P_{i-1} \), which should be equal, is presented in Figure 1. This histogram contains relative errors from all the GLAN runs used in the simulations below, 225 numbers in all. The relative errors are defined as 

\[
\frac{(P_i - P_{i-1}) - (Q_i + R_i))}{(Q_i + R_i)}. \]

It shows that the errors are all less than half a percent, and that \( Q_i + R_i \) tends to be a little smaller than \( P_i - P_{i-1} \). Figure 2 is a histogram of 225 relative errors \( (P_i - \sum_{j=1}^{i} Q_j + R_j)/(\sum_{j=1}^{i} Q_j + R_j) \).

One million sets of five normal pseudo-random variates with mean zero and variance one were used for each simulation. Each set was scaled and centered appropriately for the time points and \( \Delta \), after which each of five sets of bounds were used to determine the outcome of the simulated trial. The five types of bounds presented correspond to \( \alpha_1, \alpha_2, \alpha_3, \alpha_4, \) and \( \alpha_5 \) in Kim & DeMets (1987b), and the three tables correspond to the examples of possible timings of interim analyses given in that paper. Table 1 considers five equally spaced analyses, Table 2 unequally spaced interim analyses performed late in the trial, and Table 3 unequally spaced analyses early in the trial.

In these examples, the trapezoidal rule used a (standardized) grid size of 0.05, chosen since differences in results observed between this and 0.01 were quite small while the programs ran much faster (see Table 4 below). In each table, the first four columns were generated by LANDEM: time points, upper and lower bounds, and \( \alpha^*(t_i) - \alpha^*(t_{i-1}) \), or type 1 error to be “spent”. The following six columns alternate results from GLAN and the simulation for noncentrality parameter \( \Delta \) values of 0, 0.5, and 2.5. Note that in each table the same million simulated trials were used to produce the values for each set of bounds and value of \( \Delta \), so that the only source of variation is the type of boundary, value of \( \Delta \), or spacing of analyses.

Looking at the column headed \( \alpha^*(t_i) - \alpha^*(t_{i-1}) \) and the two columns for \( \Delta = 0 \) in the tables allows us to compare results from LANDEM, GLAN and the simulation. The agreement is good to the fourth or fifth decimal when comparing GLAN and LANDEM and to within two standard deviations for the simulated values. The columns for \( \Delta = 0.5 \) and \( \Delta = 2.5 \) allow comparison of results from GLAN and the simulation when the null hypothesis is not true, and also show good agreement.

Figure 3 provides a graphical analysis of the errors. It is a histogram of 225 differences
Figure 1: \[((P_i - P_{i-1}) - (Q_i + R_i))/(Q_i + R_i)\]; 225 internal errors in GLAN.
Figure 2: \((P_i - \sum_{j=1}^i Q_j + R_j)/(\sum_{j=1}^i Q_j + R_j)\): 225 internal errors in GLAN.
Table 1: Comparing results from **LANDEM, GLAN** and a simulation for five equally spaced interim analyses.

<table>
<thead>
<tr>
<th>time</th>
<th>LANDEM</th>
<th>Δ = 0.0</th>
<th>Δ = 0.5</th>
<th>Δ = 2.5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>lower</td>
<td>upper</td>
<td>α(ψₜ₀)</td>
<td>α(ψₜ₁)</td>
</tr>
<tr>
<td>0.2</td>
<td>-1.8684</td>
<td>4.8684</td>
<td>0.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td>0.4</td>
<td>-3.3576</td>
<td>3.3576</td>
<td>0.00079</td>
<td>0.00079</td>
</tr>
<tr>
<td>0.6</td>
<td>-2.6809</td>
<td>2.6809</td>
<td>0.00682</td>
<td>0.00682</td>
</tr>
<tr>
<td>0.8</td>
<td>-2.2903</td>
<td>2.2903</td>
<td>0.01679</td>
<td>0.01679</td>
</tr>
<tr>
<td>1.0</td>
<td>-2.0314</td>
<td>2.0314</td>
<td>0.02555</td>
<td>0.02555</td>
</tr>
<tr>
<td>0.2</td>
<td>-2.1384</td>
<td>2.1384</td>
<td>0.01477</td>
<td>0.01475</td>
</tr>
<tr>
<td>0.4</td>
<td>-2.1429</td>
<td>2.1429</td>
<td>0.01139</td>
<td>0.01139</td>
</tr>
<tr>
<td>0.6</td>
<td>-2.1102</td>
<td>2.1102</td>
<td>0.00927</td>
<td>0.00927</td>
</tr>
<tr>
<td>0.8</td>
<td>-2.3966</td>
<td>2.3966</td>
<td>0.00782</td>
<td>0.00782</td>
</tr>
<tr>
<td>1.0</td>
<td>-2.3859</td>
<td>2.3859</td>
<td>0.00676</td>
<td>0.00676</td>
</tr>
<tr>
<td>0.2</td>
<td>-2.5762</td>
<td>2.5762</td>
<td>0.01000</td>
<td>0.00999</td>
</tr>
<tr>
<td>0.4</td>
<td>-2.4920</td>
<td>2.4920</td>
<td>0.01000</td>
<td>0.00996</td>
</tr>
<tr>
<td>0.6</td>
<td>-2.1108</td>
<td>2.1108</td>
<td>0.01000</td>
<td>0.01021</td>
</tr>
<tr>
<td>0.8</td>
<td>-2.3391</td>
<td>2.3391</td>
<td>0.01000</td>
<td>0.01001</td>
</tr>
<tr>
<td>1.0</td>
<td>-2.2755</td>
<td>2.2755</td>
<td>0.01000</td>
<td>0.00996</td>
</tr>
<tr>
<td>0.2</td>
<td>-2.8431</td>
<td>2.8431</td>
<td>0.00447</td>
<td>0.00447</td>
</tr>
<tr>
<td>0.4</td>
<td>-2.5923</td>
<td>2.5923</td>
<td>0.00818</td>
<td>0.00818</td>
</tr>
<tr>
<td>0.6</td>
<td>-2.4256</td>
<td>2.4256</td>
<td>0.01059</td>
<td>0.01081</td>
</tr>
<tr>
<td>0.8</td>
<td>-2.2908</td>
<td>2.2908</td>
<td>0.01254</td>
<td>0.01256</td>
</tr>
<tr>
<td>1.0</td>
<td>-2.1750</td>
<td>2.1750</td>
<td>0.01422</td>
<td>0.01422</td>
</tr>
<tr>
<td>0.2</td>
<td>-3.0905</td>
<td>3.0905</td>
<td>0.00200</td>
<td>0.00200</td>
</tr>
<tr>
<td>0.4</td>
<td>-2.7141</td>
<td>2.7141</td>
<td>0.00600</td>
<td>0.00600</td>
</tr>
<tr>
<td>0.6</td>
<td>-2.1728</td>
<td>2.1728</td>
<td>0.01000</td>
<td>0.01022</td>
</tr>
<tr>
<td>0.8</td>
<td>-2.2798</td>
<td>2.2798</td>
<td>0.01400</td>
<td>0.01403</td>
</tr>
<tr>
<td>1.0</td>
<td>-2.1140</td>
<td>2.1140</td>
<td>0.01800</td>
<td>0.01777</td>
</tr>
</tbody>
</table>
Table 2: Comparing results from \textsc{Landem}, \textsc{GlAn} and a simulation for five unequally spaced interim analyses.

<table>
<thead>
<tr>
<th>time</th>
<th>\textsc{Landem}</th>
<th>\textsc{GlAn}</th>
<th>\textsc{GlAn}</th>
<th>\textsc{GlAn}</th>
<th>\textsc{GlAn}</th>
<th>\textsc{GlAn}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>lower</td>
<td>upper</td>
<td>sim</td>
<td>lower</td>
<td>upper</td>
<td>sim</td>
</tr>
<tr>
<td>0.3</td>
<td>-3.9282</td>
<td>3.9282</td>
<td>0.00009</td>
<td>0.00009</td>
<td>0.00007</td>
<td>0.00014</td>
</tr>
<tr>
<td>0.6</td>
<td>-2.6706</td>
<td>2.6706</td>
<td>0.00752</td>
<td>0.00752</td>
<td>0.00746</td>
<td>0.01222</td>
</tr>
<tr>
<td>0.8</td>
<td>-2.2894</td>
<td>2.2894</td>
<td>0.01679</td>
<td>0.01679</td>
<td>0.01701</td>
<td>0.02645</td>
</tr>
<tr>
<td>0.9</td>
<td>-2.1882</td>
<td>2.1882</td>
<td>0.01186</td>
<td>0.01186</td>
<td>0.01164</td>
<td>0.01812</td>
</tr>
<tr>
<td>1.0</td>
<td>-2.0773</td>
<td>2.0773</td>
<td>0.01370</td>
<td>0.01370</td>
<td>0.01369</td>
<td>0.02072</td>
</tr>
<tr>
<td>0.3</td>
<td>-2.3123</td>
<td>2.3123</td>
<td>0.02079</td>
<td>0.02076</td>
<td>0.02051</td>
<td>0.02561</td>
</tr>
<tr>
<td>0.6</td>
<td>-2.3210</td>
<td>2.3210</td>
<td>0.01464</td>
<td>0.01464</td>
<td>0.01455</td>
<td>0.02127</td>
</tr>
<tr>
<td>0.8</td>
<td>-2.3752</td>
<td>2.3752</td>
<td>0.00782</td>
<td>0.00782</td>
<td>0.00807</td>
<td>0.01247</td>
</tr>
<tr>
<td>0.9</td>
<td>-2.4357</td>
<td>2.4357</td>
<td>0.00349</td>
<td>0.00349</td>
<td>0.00351</td>
<td>0.00581</td>
</tr>
<tr>
<td>1.0</td>
<td>-2.4417</td>
<td>2.4417</td>
<td>0.00326</td>
<td>0.00326</td>
<td>0.00314</td>
<td>0.00569</td>
</tr>
<tr>
<td>0.3</td>
<td>-2.4328</td>
<td>2.4328</td>
<td>0.01500</td>
<td>0.01498</td>
<td>0.01477</td>
<td>0.01883</td>
</tr>
<tr>
<td>0.6</td>
<td>-2.3359</td>
<td>2.3359</td>
<td>0.01500</td>
<td>0.01500</td>
<td>0.01493</td>
<td>0.02192</td>
</tr>
<tr>
<td>0.8</td>
<td>-2.3228</td>
<td>2.3228</td>
<td>0.01000</td>
<td>0.01000</td>
<td>0.01027</td>
<td>0.01571</td>
</tr>
<tr>
<td>0.9</td>
<td>-2.3500</td>
<td>2.3500</td>
<td>0.00500</td>
<td>0.00500</td>
<td>0.00494</td>
<td>0.00806</td>
</tr>
<tr>
<td>1.0</td>
<td>-2.3286</td>
<td>2.3286</td>
<td>0.00500</td>
<td>0.00500</td>
<td>0.00481</td>
<td>0.00832</td>
</tr>
<tr>
<td>0.3</td>
<td>-2.6435</td>
<td>2.6435</td>
<td>0.00822</td>
<td>0.00821</td>
<td>0.00790</td>
<td>0.01067</td>
</tr>
<tr>
<td>0.6</td>
<td>-2.3677</td>
<td>2.3677</td>
<td>0.01502</td>
<td>0.01502</td>
<td>0.01495</td>
<td>0.02222</td>
</tr>
<tr>
<td>0.8</td>
<td>-2.2803</td>
<td>2.2803</td>
<td>0.01254</td>
<td>0.01254</td>
<td>0.01289</td>
<td>0.01946</td>
</tr>
<tr>
<td>0.9</td>
<td>-2.2744</td>
<td>2.2744</td>
<td>0.00691</td>
<td>0.00691</td>
<td>0.00687</td>
<td>0.01085</td>
</tr>
<tr>
<td>1.0</td>
<td>-2.2265</td>
<td>2.2265</td>
<td>0.00731</td>
<td>0.00731</td>
<td>0.00714</td>
<td>0.01168</td>
</tr>
<tr>
<td>0.3</td>
<td>-2.8412</td>
<td>2.8412</td>
<td>0.00450</td>
<td>0.00449</td>
<td>0.00427</td>
<td>0.00604</td>
</tr>
<tr>
<td>0.6</td>
<td>-2.4267</td>
<td>2.4267</td>
<td>0.01350</td>
<td>0.01350</td>
<td>0.01343</td>
<td>0.02034</td>
</tr>
<tr>
<td>0.8</td>
<td>-2.2727</td>
<td>2.2727</td>
<td>0.01400</td>
<td>0.01400</td>
<td>0.01430</td>
<td>0.02173</td>
</tr>
<tr>
<td>0.9</td>
<td>-2.2358</td>
<td>2.2358</td>
<td>0.00850</td>
<td>0.00850</td>
<td>0.00840</td>
<td>0.01317</td>
</tr>
<tr>
<td>1.0</td>
<td>-2.1625</td>
<td>2.1625</td>
<td>0.00950</td>
<td>0.00950</td>
<td>0.00944</td>
<td>0.01482</td>
</tr>
</tbody>
</table>
Table 3: Comparing results from **LANDEM**, **GLAN** and a simulation for five unequally spaced interim analyses.

<table>
<thead>
<tr>
<th>time</th>
<th><strong>LANDEM</strong></th>
<th>(\Delta = 0.0)</th>
<th>(\Delta = 0.5)</th>
<th>(\Delta = 2.5)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>lower</td>
<td>upper</td>
<td>GLAN sim</td>
<td>GLAN sim</td>
</tr>
<tr>
<td>0.1</td>
<td>-6.8742</td>
<td>6.8742</td>
<td>0.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td>0.2</td>
<td>-4.8608</td>
<td>4.8608</td>
<td>0.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td>0.3</td>
<td>-3.9298</td>
<td>3.9298</td>
<td>0.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td>0.6</td>
<td>-2.6706</td>
<td>2.6706</td>
<td>0.00008</td>
<td>0.00008</td>
</tr>
<tr>
<td>1.0</td>
<td>-1.9815</td>
<td>1.9815</td>
<td>0.04234</td>
<td>0.01234</td>
</tr>
<tr>
<td></td>
<td>0.017055</td>
<td>2.6555</td>
<td>0.00793</td>
<td>0.00792</td>
</tr>
<tr>
<td>0.2</td>
<td>-2.6233</td>
<td>2.6233</td>
<td>0.00684</td>
<td>0.00684</td>
</tr>
<tr>
<td>0.3</td>
<td>-2.5896</td>
<td>2.5896</td>
<td>0.00602</td>
<td>0.00602</td>
</tr>
<tr>
<td>0.6</td>
<td>-2.3488</td>
<td>2.3488</td>
<td>0.01464</td>
<td>0.01464</td>
</tr>
<tr>
<td>1.0</td>
<td>-2.2792</td>
<td>2.2792</td>
<td>0.01457</td>
<td>0.01457</td>
</tr>
<tr>
<td></td>
<td>0.01874</td>
<td>2.8074</td>
<td>0.00500</td>
<td>0.00499</td>
</tr>
<tr>
<td>0.2</td>
<td>-2.7103</td>
<td>2.7103</td>
<td>0.00500</td>
<td>0.00500</td>
</tr>
<tr>
<td>0.3</td>
<td>-2.6724</td>
<td>2.6724</td>
<td>0.00500</td>
<td>0.00500</td>
</tr>
<tr>
<td>0.6</td>
<td>-2.3565</td>
<td>2.3565</td>
<td>0.01500</td>
<td>0.01500</td>
</tr>
<tr>
<td>1.0</td>
<td>-2.1838</td>
<td>2.1838</td>
<td>0.02000</td>
<td>0.02000</td>
</tr>
<tr>
<td></td>
<td>0.01596</td>
<td>3.1596</td>
<td>0.00158</td>
<td>0.00158</td>
</tr>
<tr>
<td>0.2</td>
<td>-2.9412</td>
<td>2.9412</td>
<td>0.00289</td>
<td>0.00289</td>
</tr>
<tr>
<td>0.3</td>
<td>-2.8007</td>
<td>2.8007</td>
<td>0.00374</td>
<td>0.00374</td>
</tr>
<tr>
<td>0.6</td>
<td>-2.3775</td>
<td>2.3775</td>
<td>0.01502</td>
<td>0.01502</td>
</tr>
<tr>
<td>1.0</td>
<td>-2.0954</td>
<td>2.0954</td>
<td>0.02676</td>
<td>0.02677</td>
</tr>
<tr>
<td></td>
<td>0.3.4809</td>
<td>3.4809</td>
<td>0.00050</td>
<td>0.00050</td>
</tr>
<tr>
<td>0.2</td>
<td>-3.1523</td>
<td>3.1523</td>
<td>0.00150</td>
<td>0.00150</td>
</tr>
<tr>
<td>0.3</td>
<td>-2.9508</td>
<td>2.9508</td>
<td>0.00250</td>
<td>0.00250</td>
</tr>
<tr>
<td>0.6</td>
<td>-2.4317</td>
<td>2.4317</td>
<td>0.01350</td>
<td>0.01350</td>
</tr>
<tr>
<td>1.0</td>
<td>-2.0462</td>
<td>2.0462</td>
<td>0.03200</td>
<td>0.03200</td>
</tr>
</tbody>
</table>
Figure 3: A histogram of 225 discrepancies between values computed in GLAN and results from the simulation.

between probabilities computed by GLAN and estimated by the simulation. Keeping in mind that these are not strictly independent, each difference was divided by $\sqrt{P_i \times (1 - P_i)}$, as computed by GLAN, to put them on roughly the same scale. The remaining variability combines contributions from the simulation and from errors in computation. The contribution of the simulation to this variance is on the order of $10^{-6}$. Figure 3 reveals a distribution of errors with mean $-0.00029$, variance $2.2 \times e^{-9}$, possibly a little skewed to the left.

As a check against other results in the literature, GLAN was run for 20 equally spaced interim analyses and two-sided, $\pm 1.96$ bounds. These numbers were compared against values tabulated in Armitage et al. (1969). Table 4 contains numbers reported by Armitage et al., along with corresponding numbers from GLAN for both the default grid size (0.05) and from a version in which the grid size was set to 0.01. There are two columns from Armitage et al., one labeled Q and one labeled S, which represent the probability of absorption (exceeding the bounds at the $n^{th}$ analysis) calculated by quadrature and by simulation, respectively.
Table 4: Comparing results from GLAN and Armitage et al.

<table>
<thead>
<tr>
<th></th>
<th>GLAN</th>
<th>Armitage et al.</th>
<th>Grid = .01</th>
<th>Grid = .05</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>Q</td>
<td>S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.05000</td>
<td>0.0545</td>
<td>0.05000</td>
<td>0.05000</td>
</tr>
<tr>
<td>2</td>
<td>0.08312</td>
<td>0.0885</td>
<td>0.08311</td>
<td>0.08311</td>
</tr>
<tr>
<td>3</td>
<td>0.10726</td>
<td>0.1115</td>
<td>0.10725</td>
<td>0.10724</td>
</tr>
<tr>
<td>4</td>
<td>0.12617</td>
<td>0.1260</td>
<td>0.12616</td>
<td>0.12615</td>
</tr>
<tr>
<td>5</td>
<td>0.14169</td>
<td>0.1420</td>
<td>0.14168</td>
<td>0.14166</td>
</tr>
<tr>
<td>10</td>
<td>0.19336</td>
<td>0.1925</td>
<td>0.19335</td>
<td>0.19331</td>
</tr>
<tr>
<td>15</td>
<td>0.22509</td>
<td>0.2195</td>
<td>0.22508</td>
<td>0.22504</td>
</tr>
<tr>
<td>20</td>
<td>0.24791</td>
<td>0.2455</td>
<td>0.24790</td>
<td>0.24785</td>
</tr>
</tbody>
</table>

The last two columns are the corresponding results from GLAN with grid size 0.05 and 0.01.

Armitage et al. used a grid of 0.01 for their results, and the results from the GLAN run with a grid size of 0.01 are very close. The simulation Armitage et al. performed involved 2000 repetitions, and so is not as precise as the simulation results in the previous tables—not reliable beyond the third decimal. To this level of accuracy, values from GLAN with a grid of 0.05 are quite adequate.

D Routines specific to LANDEM

Program GSELD, calling subroutine LANDEM.

Given only the past history of interim analyses, this program computes the next boundary for group sequential tests of the hypothesis that the mean of the process is zero.

It follows the "original" intention of Lan & DeMets (1983) in that the number and the times of interim analyses need not be specified in advance, in contrast to the predetermined group sequential by Pocock (1977) or by O'Brien & Fleming (1979). In addition users familiar with Lan & DeMets (1980) may specify separate time scales for calculation of type I error to be spent and bounds.


n is the number of interim analyses
delta is the true mean of process: drift of brownian motion
c side indicates one or two-sided symmetric boundary
c use indicates use function for upper boundary (see Kim & DeMets)
c alpha is the desired significance level
c ti is the information times of interim analyses.
c en is the number of events
c pb is the significance level specified by the use function.
c zb is the standardized upper bound.
za is the standardized lower bound.

strun is the truncation point for integration limits

ndim specifies maximum number of interim analyses.

ldin and ldot are input and output routines.

program gslid

integer ndim
parameter(ndim=50)

double precision ti(ndim),en(ndim),pn(ndim),pd(ndim),
za(ndim),zb(ndim),alpha,side,delta,strun

integer n,use,ieinf

call ldin(n,delma,alp,ti,en,side,use,ieinf,strun)
call ldoma(n,delma,alp,ti,en,side,use,strun, pn,pd,za,zb)
call ldot(n,alp,use,side,ieinf,ti,en,pd,delma,za,zb)

stop
end

subroutine ldin(n,delma,alp,ti,en,side,use,ieinf,strun)

Input values for ldom interactively or non-interactively.

double precision delma,alp,ti(en),en(side),
strun

integer n,use,ieinf

integer i,nn,inside,inter,strun,ibad
character char

Refer to subroutine alphas for use function codes.

Control for input from screen vs. input from a file.
write(6,*) ' Is this an interactive session? (i=yes,0=no)',
read(6,5) char

5 format(10a)
if ((a. eq. 'i').or.(a. eq. 'y')) then
  inter = 1
else if ((a. eq. '0').or.(a. eq. 'n')) then
  inter = 0
else
  Invalid responses are assumed to be interactive!
  write(6,*) ' invalid response.'
  inter = 1
end if
write(6,*) ' interactive = ', inter

Total type I error probability.
if (inter.eq.1) write(6,*) ' Overall significance level?',
  '(>0 and <=1)',
read(6,9) alpha
if ((alpha <= 0.0) .or. (alpha .gt. 1.0)) then
  Out of range sets alpha = 0.05!
  write(6,*) ' invalid response.'
  write(6,*) ' Alpha must be > 0 and <= 1.'
  alpha = 0.05
  write(6,*) ' Alpha has been set to 0.05.'
end if
write(6,10) alpha
10 format(' alpha = ',f5.3)

Upper and lower bounds, or upper only?
if (inter.eq.1) write(6,*) ' One(1)- or two(2)-sided symmetric?',
read(6,12) char

12 format(10a)
if ((a. eq. '1').or.(a. eq. 'o')) then
  iside = 1
else if ((a. eq. '2').or.(a. eq. 't')) then
  iside = 2
else
  Invalid responses are assumed to be two-sided!
  write(6,*) ' invalid response.'
inside = 2
end if
write(6,'*') ' ,inside, ' -sided test'

Alpha-star use functions can be edited in alpha.f.
if (int.eq.1) then
  write(6,'*') ' Use function? (1-5)'
  write(6,'*') (1) O'Brien-Fleming type'
  write(6,'*') (2) Pocock type'
  write(6,'*') (3) alpha * t'
  write(6,'*') (4) alpha * t^1.5'
  write(6,'*') (5) alpha * t^2'
end if
read(5,*) use
write(6,'*') ' Use function alpha-star'; use

As currently written, the program only truncates by reporting
any bound larger than the truncation point as 3.5 without
any adjustment to the probabilities. Since this is misleading,
the code for this option has been removed.

Does the design include a truncation point?
if (int.eq.1) write(6,'*') ' Do you wish to truncate the',
  ' standardized bounds? (1=yes, 0=no)'
read(5,15) char
format(10a)
if ((char.eq.'y') .or. (char.eq.'y')) then
  itrun = 1
else if ((char.eq.'y') .or. (char.eq.'y')) then
  itrun = 0
else
  Invalid responses are assumed to mean no truncation!
write(6,'*') ' Invalid response.'
  itrun = 0
end if

if (itrun.eq.1) then
  if (int.eq.1) write(6,'*') ' Enter truncation point: '
  read(5,*) ztrun
write(6,20) ztrun
format(' Bounds will be truncated at ',f5.2)
else
  write(6,'*') ' Bounds will not be truncated.'
end if

How many interim analyses have already been done?
if (int.eq.1) write(6,'*') ' Number of interim analyses',
  ' in the past (0 if this is the first):'
read(5,*) nml
if (nml.eq.0) then
  write(6,'*') ' First analysis.'
else
  write(6,'*') nml, ' previous analyses.'
end if

Total number of analyses, including the current one.
n = nml + 1

If this is not the first analyses, enter previous times.
if (n.ne.1) then
  if (int.eq.1) write(6,'*') ' Enter times (or information',
    ' fractions > O and < 1) in the past: '
  read(5,*) (ti(i), i=1,nml)
write(6,30) (ti(i), i=1,nml)
format(' Previous times (or info fractions) are ',
  28(f5.3,1x))
end if

User enters the current time.
if (int.eq.1) write(6,'*') ' Time or info fraction (<= 1)',
  ' of the current interim analysis?'
read(5,*) ti(n)
write(6,40) ti(n)
format(' Current time (or information fraction) is ',f5.3)
c Verify range and ordering of times
   ibad = 0
   if (ti(1).le.0.0) ibad = 1
   do 43 i=2,n
      if (ti(i).le.ti(i-1)) ibad = 1
   continue
   if (ti(n).gt.1.0) ibad = 1
   if (ibad.eq.1) then
      c Times out of range or order is fatal.
      write(6,*) ' Error in input! Please verify:
      write(6,*) ' Times must be strictly increasing in',
      write(6,*) ' (0,1]. Aborting.'
      stop
   end if
   c Will exact or estimated information be used?
   if(int.eq.1) write(6,*) ' Do you wish to input',
      ' the exact or estimated information? (e.g.,',
      ' number of patients or number of events,)
      ' as in Lan & DeMets 89?) ',
      ' (y/n/e)'
   end if
   read(6,45) char
   c format(10a)
   if ((char.eq.'y') .or. (char.eq.'e')) then
      iieinf = 1
   else if ((char.eq.'n')) then
      iieinf = 0
   else
      invalid responses are assumed to mean time only!
      write(6,*) ' invalid response.'
      iieinf = 0
   end if
   if (iieinf.eq.1) write(6,*) ' Entering information.'
   if iieinf.eq.1 then
      if (int.eq.1) then
         c If information is being used, enter.
         if (n.ne.1) then
            c If information is being used, enter.
            if (int.eq.1) then
               c If information is being used, enter.
               if (n.ne.1) then
                  iieinf = 1
               end if
            end if
         end if
      end if
   end if
   c Verify range and ordering of information
   ibad = 0
   if (en(1).le.0.0) ibad = 1
   do 70 i=2,n
      if (en(i).le.en(i-1)) ibad = 1
   continue
   if (ibad.eq.1) then
      c Information <=0 or out of order causes them to be ignored.
      write(6,*) ' Error in input! Please verify:
      write(6,*) ' Information must be strictly increasing',
      ' and > 0. Input is ignored, the analysis',
      ' proceeds using times (info frac) only.'
      do 90 i=1,n
         en(i) = ti(i)
      continue
   end if
   c else
   c Time and information are set equal unless
   c otherwise specified.
   do 100 i=1,n
      en(i) = ti(i)
   continue
   c end if
Only noncentrality of 0 is currently implemented.
write(6,*)
write(6,*) ' Delta will be taken to be zero.'
delta = 0.00

Convert integer indicator to double precision # of sides
if (side .eq. 1) then
  side = 1.00
else
  side = 2.00
end if

write(6,*)
return
end

subroutine ldout(n, alpha, use, side, ierror, ti, en, pd, delta, za, zb)

Output values from lancem.

double precision alpha, side, ti(+), en(+), pd(+), delta, za(+), zb(+)
integer n, use, ierror

double precision pv
integer i

Print out results.
if (side .eq. 1.00) then
  write(6,10)
  write(6,20) n, alpha, use
  format('input this program generates one-sided boundaries.')
  format('n = ', 'i2' / 'alpha = ', 'f6.3' / use function = ', 'i1')
else
  write(6,30)
  write(6,40) n, alpha, use
  format('input this program generates two-sided',
        'symmetric boundaries.')
  format('n = ', 'i2' / 'alpha = ', 'f6.3'
        'use function for the lower boundary = ', 'i1'
        'use function for the upper boundary = ', 'i1')
end if

if (ierror .eq. 0) then
  write(6,*) ' Time Bounds ',
  'alpha(i)-alpha(i-1) cum alpha'
pv = 0.00
  do 100 i=1,n
    pv = pv + pd(i)
    write(6,150) ti(i), en(i), za(i), zb(i), pd(i), pv
    continue
  endformat(4x, 'f5.2, 12x, 'f7.4, 4x, 'f7.4, 9x, 'f7.5, 9x, 'f7.5)
else
  write(6,*) ' Time Information Bounds ',
  'alpha(i)-alpha(i-1) cum alpha'
pv = 0.00
  do 200 i=1,n
    pv = pv + pd(i)
    write(6,250) ti(i), en(i), za(i), zb(i), pd(i), pv
    continue
  endformat(4x, 'f5.2, 5x, 'f7.2, 4x, 'f7.4, 2x, 'f7.4, 7x, 'f7.5, 7x, 'f7.5)
end if

return
end

subroutine lancem(nn, delta, alpha, ti, en, side, use, strun, 
  pd, za, zb)

Subroutine to calculate boundaries corresponding to
particular type I error spending rates and times of
interim analyses, per Lan & DeMets 83.

Input Variables:
nn is the number of interim analyses.
delta is the true mean of the process, drift of br.mot.
delta is set to zero, and so does not affect program.
We delta MAY NOT be changed without adding code!
alpha is the total type I error
ti is the information times
en are the numbers of events
side indicates one or two sided boundaries.
use indicates type I error spending function.
strun is the user selected truncation on integration limits.

Returned variables:
time is the expected first exit time.
pi is a vector of nominal exit probabilities.
pd(i) = pn(i)-pn(i-1)
za is the vector of lower standardized boundaries
zb is the vector of upper standardized boundaries

Local variables:
stdv is the standard deviation of the process increment.
stproc in the standard deviation of the process.
last is the grid of the joint density at the last analysis.
nints is the number of intervals for numerical integration.
ya is the vector of lower integration limits.
yb is the vector of upper integration limits.
zinf is "negative infinity" for the program

Program parameters:
N is the grid size for numerical integration by trapezoidal rule.
lNN is the maximum dimension for last()
maxnn is the maximum number of interim analyses.

Subroutines and functions called directly:
znorm is the inverse standard normal cdf.
alphas computes pn() according to specified use function.
search finds second and later integration bounds
first computes joint density of process at first analysis.
other computes joint density of process at second and later.

double precision za(*),zb(*),alpha,ti(*),en(*),pn(*),pd(*),
side,delta,strun,tol
integer nn,use

double precision h
integer lNN,maxnn
parameter(maxnn=50,lNN=5000,h=0.08d0)

double precision stdv(maxnn),stproc(maxnn),last(lNN),
y(a(maxnn)),yb(maxnn),zninf
integer i,nints(maxnn)
double precision znorm

zninf = -3.4d0
write(6,10) zninf
10 format(' Negative infinity set to ',f7.3,
if (strun.le.0.d0) strun = -zninf

tolerance for type I error to spend after 1st analysis.
tol = 1.d-07

calculate probabilities according to use function.
call alphas(use,nn,alphan,side,ti,pn,pd)
calculate standard deviations of increments and process.
call sd(nn,sn,stdv,stproc)

begin loop calculating boundaries.
calculate probabilities according to use function.
call alphas(use,nn,alphan,side,ti,pn,pd)
calculate standard deviations of increments and process.
call sd(nn,sn,stdv,stproc)

begin loop calculating boundaries.
direct calculations can be made for first analysis.
c
check type I error to spend
if (pd(1) .lt. 0.d0) or (pd(1) .gt. 1.d0) then
write(*,*) 'Error in spending function.'
pd(1) = min(1.d0,pd(1))
pd(1) = max(0.d0,pd(1))

29
end if

if (pd(i) .eq. 0.0d0) then
  zb(i) = min(strun,-zminf)
  yb(i) = zb(i)*stdv(i)
else
  Spending probability is one (or more)
  if (pd(i) .eq. 1.0d0) then
    zb(i) = 0.0d0
    yb(i) = zb(i)*stdv(i)
  else
    First bound based on normal distribution.
    zb(i) = min(zerom(1.0d0-(pd(i)/side)),strun)
    yb(i) = zb(i)*stdv(i)
  end if

if (side.eq.1.0d0) then
  za(i) = zminf
  ya(i) = za(i)*stdv(i)
else
  za(i) = -zb(i)
  ya(i) = -yb(i)
end if

Number of intervals for numerical integration.
nints(i) = idint((yb(i)-ya(i))/(h*stdv(i))) + 1

Calculations for second and later analyses.
do 100 i=2,nn
  Calculate joint density for use in next step.
  if (i.eq.2) call first(ya(i),yb(i),h, stdv(i), last, nints)
  Check type I error to spend
  if ((pd(i) .lt. 0.0d0) .or. (pd(i) .gt. 1.0d0)) then
    write(6,'* Error in spending function.')
    pd(i) = min(1.0d0,pd(i))
    pd(i) = max(0.0d0,pd(i))
  end if

if (pd(i) .lt. tol) then
  zb(i) = min(strun,-zminf)
  yb(i) = zb(i)*stdv(i)
else
  Spending probability is one (or more)
  if (pd(i) .eq. 1.0d0) then
    zb(i) = 0.0d0
    yb(i) = zb(i)*stdv(i)
  else
    Bounds are found using a search starting at the
    bound from the previous analysis.
    call search(last,nints,i,h,pd(i)/side, stdv(i),ya, yb)
    yb(i) = min(yb(i), strun*sdproc(i))
    zb(i) = yb(i)/sdproc(i)
  end if

if (side.eq.1.0d0) then
  ya(i) = zminf*sdproc(i)
  za(i) = zminf
else
  ya(i) = -yb(i)
  za(i) = -zb(i)
end if

Number of intervals for numerical integration.
nints(i) = idint((yb(i)-ya(i))/(h*stdv(i))) + 1

Calculate joint density for use in next step.
if (i.ne.n) call other(ya,yb,i, stdv(i),h, last, nints)
100 continue
c
return
end

c subroutine nd(nn,t, sdincr,sdproc)
c Standard deviations are calculated. Users may
modify for particular applications, eg rescaling
by std(v(i)/std(v(nn)):
c sdincr(i) = dsqrt(t(i)/t(nn))
c sdproc(i) = sdincr(i)
c
double precision t(*),sdincr(*),sdproc(*)
integer nn

integer i

loop for standard deviations
do 100 i=1,nn
Standard deviation for first analysis:
if (i.eq.1) then
    sdincr(i) = dsqrt(t(i)/t(nn))
    sdproc(i) = sdincr(i)
else
Standard deviation for second and later analyses:
    sdincr(i) = dsqrt((t(i)-t(i-1))/t(nn))
    sdproc(i) = dsqrt(t(i)/t(nn))
end if
100 continue
return
end

subroutine alphas(use,nn,alpha,sidel,ti,pn,pd)
c This use function, or type I error spending rate,
corresponds to the alpha 1 star in Lan & DeMets(1983).
c Alpha 2 star in Lan & DeMets (1983).
c Alpha 3 star in Lan & DeMets (1983).
c Alpha 4 star in Kim & DeMets (1987).
c Alpha 5 star in Kim & DeMets (1987).
c
Input variables:
c use indicates type I error spending rate function.
c nn is the number of interim analyses so far.
c alpha is the desired overall size.
c side is the number of sides for the test.
c ti is the vector of information times.
c
Returned variables:
c pn is the vector of type I error spent at each analysis.
c pd is the differences in type I error spent between analyses.
c
Local variables:
c e is the base for natural logarithms.
c
Subroutines and functions called directly:
c pnorm is the standard normal cdf.
c znorm is the inverse standard normal cdf.
c
double precision alpha,sidel,ti(*),pn(*),pd(*)
integer use,nn
c
integer i
do 50 i=1,nn
    if (use .eq. 1) then
        pn(i) = 2.0d0*
        (1.0d0-pnorm(znorm(1.0d0-(alpha/sidel)/2.0d0)/dsqrt(ti(i)))))
50 continue
else if (use.eq.2) then
  pn(i)=(alpha+side)*log(1.d0+(e-1.d0)*ti(i))
else if (use.eq.3) then
  pn(i)=(alpha+side)*ti(i)
else if (use.eq.4) then
  pn(i)=(alpha+side)*(ti(i)**1.5d0)
else if (use.eq.5) then
  pn(i)=(alpha+side)*(ti(i)**2.0d0)
else
  write(6,*) ' Warning: invalid use function.'
end if

pd is the change in type I error spent
if (eq.i) then
  pd(i) = pn(1)
else
  pd(i) = pn(i) - pn(i-1)
end if

if (pd(i) .lt. tol) then
  write(6,*), ' Type I error spent too small, analysis', i
  write(6,*) ' Zero used as approximation for ', pd(i)
  pd(i) = 0.d0
end if

50 continue
return
end

subroutine search(last,nints,i,h,pd,stdv,ya,yb)
C
C A naive searching algorithm. Users may want to
C substitute a better routine. This starts at the previous
C boundary, moves in the direction of the current boundary,
C changes direction and reduces step size when target is
C overstepped. Steps when probability of estimate is within
C eps of target probability.
C
C Input variables:
C last is joint density from the previous analysis.
C nints is the number of intervals for numerical integration.
C i is the number of the current analysis.
C h is the grid size.
C pd is the target probability.
C stdv is the standard deviation of the process increment.
C
C Returned variables:
C ya is the vector of lower integration limits.
C yb is the vector of upper integration limits.
C
C Local variables:
C uppr is the estimate for yb(i).
C q is the probability associated with uppr.
C del is the searching step size.
C eps is the tolerance for stopping.
C
C Subroutines and functions called directly:
C qpos computes tail probabilities.
C
double precision last(*),ya(*),yb(*),h,pd,stdv
integer i,nints(*)

double precision uppr,del,eps,q

double precision qpos

c Initialize tolerance for abs(q-pd).
eps = 1.d-7

c Initialize step size.
del = 10.0d0

c Initialize estimates at previous integration limit.

32
uppr = yb(i-1)
q = qpos(uppr, last, nints(i-1), ya(i-1), yb(i-1), h, stdv)

Begin searching.
10 continue
C
C If q and pd are nearly equal, set yb(i) and return.
if (dabs(q-pd).le.epe) then
yb(i) = uppr
return
C
If q is too large, start increasing uppr by steps.
else if (q.gt.(pd+eps)) then
C
Reduce step size by a factor of 10.
del = del/10.d0
C
Increase uppr by del and check whether q is near pd.
do 30 j=1,50
uppr = uppr + del
q = qpos(uppr, last, nints(i-1), ya(i-1), yb(i-1), h, stdv)
if (q.le.(pd+eps)) go to 10
30 continue
C
If many iterations do not converge, print warning.
if (mod(j,10).eq.0) write(6,90) i

After many steps, abort.
write(6,100) i
return
C
C If q is too small, follow analogous procedure.
else if (q.lt.(pd-eps)) then
40 del = del/10.d0
C
Increase uppr by del and check whether q is near pd.
do 80 j=1,50
uppr = uppr - del
q = qpos(uppr, last, nints(i-1), ya(i-1), yb(i-1), h, stdv)
if (q.ge.(pd-eps)) go to 10
if (mod(j,10).eq.0) write(6,90) i
80 continue
write(6,100) i
return
C
C Format('Large change in bounds, possible error analysis',i4)
C Format('Error in search: not converging. Abort analysis',i4)
C
C double precision function qpos(xq, last, nints, yaml, ybml, h, stdv)
C
Routine calculates tail probability by numerical integration.
C
Input variables:
xq is the argument for evaluation
last is the vector of density values from the previous step.
nints is the number of intervals for numerical integration.
yaml is the lower integration limit from the previous step.
ybml is the upper integration limit from the previous step.
h is the grid size.
stdv is the standard deviation of the process increment.
C
Local variables:
fun is a temporary vector of function values.
grid is the step, yaml + h(j-1)
C
Program parameters:
mwork dimensions fun.
C
Subroutines and functions called directly:
 pcm is the standard normal cdf.
 trap is a trapezoidal rule numerical integration.
C
double precision xq, last(*), yaml, ybml, h, stdv
integer nints

33
E  Routines specific to GLAN

Program GSBLAN, using function GLAN

This program acts as a driver for glan. Subroutine "set" is called to input desired sequential boundaries and times of interim analyses. Subroutine "glan" calculates associated probabilities, "output" prints a table of results.

n is the number of interim analyses.
t is the vector of analysis time on (0,1] scale.
za is the standardized lower boundary.
zb is the standardized upper boundary.
delta = true mean of process, drift of brownian motion.
pstr(i) is the prob of reaching i-th analysis and stopping.
qpos(i) is the prob of reaching i-th and exceeding upper.
qneg(i) is the prob of reaching i-th and exceeding lower.
est is the expected stopping time.
pb is the total prob of rejecting, sum(i,n)(qpos+qneg)

program geblan

integer ndim
parameter(ndim=25)

double precision za(ndim),zb(ndim),pstr(ndim),qpos(ndim),
        qneg(ndim),t(ndim),pb,delta,est
character char
integer n

call glin(za,zb,t,n,delta)
continue

call glan(n,t,za,zb,delta, pstr,qpos,qneg,est,pb)
call glout(n,pb,delta,est,t,za,zb,pstr,qpos,qneg)

The following lines add the option of repeating the computations using a new value for the drift parameter.

write(6,*) ' Do you wish to recompute using a ',
         ' new drift parameter (delta) (1=yes,0=no)?'
read(5,60) char

format(10a1)
if ((char.eq.'1') .or. (char.eq.'y')) then
write(5,*), write(5,*), write(6,*), write(5,*), write(6,*), write(6,*)
write(6,*), Enter new drift parameter:
read(5,*) delta
write(6,*), Recomputing with delta = ', delta
go to 50
else if ((char.eq.'O').or.(char.eq.'n')) then
    write(6,*), 'Done.'
else
    Invalid responses are assumed to be 'no'
    write(6,*), 'Invalid response.'
end if
write(6,*), 'Goodbye.'
write(6,*)

stop

end

subroutine glin(za,zb,t,n,delta)

Values for input to glin.

double precision za(*),zb(*),t(*),delta
integer n

double precision zinf
integer idel,iside,lsymm,ifile,ibad
character,char

Uncomment this line if you want to input bounds
from a file instead of interactively
*** open(unit=20,file='userin')

Program negative infinity for one sided bounds.
zinf = -0.000

Control for input from screen vs. input from a file.
write(6,*), 'Is this an interactive session? (1=yes,0=no)'
read(5,5) char
format(10a1)
if ((char.eq.'Y').or.(char.eq.'y')) then
    inter = 1
else if ((char.eq.'O').or.(char.eq.'n')) then
    inter = 0
else
    Invalid responses are assumed to be interactive!
    write(6,*), 'Invalid response.'
    inter = 1
end if
write(6,*), 'interact = ', inter

# of looks should be < ndim in calling routine.
if (inter.eq.1) write(6,*), Number of interim analyses:
read(5,*), n
write(6,*), n, interim analyses.

One time for each look.
if (inter.eq.1) write(6,*), Times of interim analyses:
read(5,*), (t(i),i=1,n)
write(6,10) (t(i),i=1,n)
format(1,12F8.3,1x)

Verify range and ordering of times
ibad = 0
if (t(i).le.0.0) ibad = 1
do 13 ibad = 2,n
if (t(i).le.t(i-1)) ibad = 1
13 continue
if (ibad.eq.1) then
    Times out of range or order is fatal.
    write(6,*), 'Error in input! Please verify!'
write(6,*), 'Times must be strictly increasing in',
      ' (0,1]. Aborting.'
    stop

35
User enters noncentrality parameter.
if (inter.eq.1) write(6,*) 'Do you wish to use drift parameter?',
      (Delta) other than zero? (i=yes,0=no)
read(5,15) char
if ((char.eq.'1').or.(char.eq.'y')) then
   idel = 1
else if ((char.eq.'0').or.(char.eq.'n')) then
   idel = 0
else
   Invalid responses are assumed to be no drift.
   write(6,*) 'Invalid response.'
   idel = 0
end if

Default is zero.
if (idel.eq.1) then
   if (inter.eq.1) write(6,*) 'Enter noncentrality parameter: '
      read(6,*) delta
else
   delta = 0.0d0
end if
write(6,*) 'Delta = ',delta

User enters choice for one or two sided bounds.
if (inter.eq.1) write(6,*) 'One(1)- or two(2)-sided?
read(5,20) char
20 format(10a)
if ((char.eq.'1').or.(char.eq.'o')) then
   iside = 1
else if ((char.eq.'2').or.(char.eq.'t')) then
   iside = 2
else
   Invalid responses are assumed to be two-sided.
   write(6,*) 'Invalid response.'
   iside = 2
end if
write(6,*) 'iside,-sided test'

Two sided bounds may be symmetric or asymmetric.
if (iside.eq.2) then
   if (inter.eq.1) write(6,*) 'Symmetric bounds? (i=yes,0=no)
      read(5,25) char
25 format(10a)
if ((char.eq.'1').or.(char.eq.'y')) then
   isym = 1
else if ((char.eq.'0').or.(char.eq.'n')) then
   isym = 0
else
   Invalid responses are assumed to be symmetric.
   write(6,*) 'Invalid response.'
   isym = 1
end if
if (isym.eq.0) then
   write(6,*) 'Two sided asymmetric bounds.'
else
   write(6,*) 'Two sided symmetric bounds.'
end if
else
   isym = 1
   write(6,*) 'One sided bounds.'
end if

Users may input bounds to a file, here 'userin', instead
of inputting from the screen or standard input. This
will usually require modification based on local system.
if (inter.eq.1) write(6,*) 'Will bounds be input from a file?
   file? (i=yes,0=no)
read(5,35) char
35 format(10a)
if ((char.eq.'1').or.(char.eq.'y')) then
   ifile = 1
else if ((char.eq.'0').or.(char.eq.'n')) then
   ifile = 0
else
    Invalid responses are assumed to be no.
    write(6,*) ' Invalid response.'
    ifile = 0
end if

if (ifile.eq.0) then
    if (inter.eq.1) write(6,*) ' Enter upper bounds in',
    ' standardized form: ',
    read(5,*) (zb(i),i=1,n)
    if (iside.eq.2) then
        if (isym.eq.0) then
            if (inter.eq.1) write(6,*) ' Enter lower bounds in',
            ' standardized form: ',
            read(5,*) (za(i),i=1,n)
        else
            do 30 i=1,n
               za(i) = -zb(i)
            continue
        end if
    else
        do 40 i=1,n
           za(i) = zninf
        continue
    end if
else
    write(6,*) ' Bounds being entered from a file.'
    read(20,*) (zb(i),i=1,n)
    if (iside.eq.2) then
        if (isym.eq.0) then
            read(20,*) (za(i),i=1,n)
        else
            do 50 i=1,n
               za(i) = -zb(i)
            continue
        end if
    else
        do 60 i=1,n
           za(i) = zninf
        continue
    end if
end if
write(6,*) ' Bounds entered.'

return
end

subroutine glout(n,pb,delta,est,t,za,zb,pstr,qpos,qneg)

Print out results from glan

double precision pb,delta,est,t(*),za(*),zb(*),
        pstr(*),qpos(*),qneg(*)
integer n

double precision qplsr,qrsum,pdif
integer i

write(6,10) n,delta
10 format(/8x,'n =',i3,', delta =',f6.4)
write(6,20)
20 format('/ look',5x,'time',5x,'lower',4x,'upper',4x,
      'alpha(i)-alpha(i-1)',3x,'cum alpha/
      qplsr',5x,'qrsum/
      pdif',5x,'qplsr',5x,'pstr',5x,'qrsum/
      qrsum = 0.00
    do 100 i=1,n
       qplsr = qpos(i)+qneg(i)
       qrsum = qrsum + qplsr
       if (i.eq.1) then
          pdif = pstr(i)
       else

37
\begin{verbatim}

end

subroutine error1(nn, ndim)
write(6,10) nn, ndim
format(5x,'glan: No. of analyses too large, n = ',i5,','',i5)
return
end

subroutine error2(i, lmn)
write(6,10) i, lmn
format(5x,'glan: Bounds too wide, analysis',i3,'

Coman alpha level and other probabilities for the number of

tests of hypotheses. Method used is a modification of Armitage,

McPherson and Rowe. Consistent with Lan & DeMets 83, does not

assume equally spaced information times.

Input variables:

- \( n \) is the number of analyses.
- \( t \) is the vector of analysis time point on \((0,1)\).
- \( z_b \) is the vector of lower bounds (standardized).

Returned variables:

- \( \text{petr}(i) \) is the prob of reaching \( i \)th analysis and stopping.
- \( \text{qpos}(i) \) is the prob of reaching \( i \)th and exceeding upper.
- \( \text{qneg}(i) \) is the prob of reaching \( i \)th and exceeding lower.
- \( \text{est} \) is the expected stopping time.
- \( \text{pb} \) is the total type I error used.

Local variables:

- \( f_n \) is the joint density for the current analysis.
- \( l_n \) is the joint density at the last analysis.
- \( y_b \) is the vector of upper integration limits.
- \( y_a \) is the vector of lower integration limits.
- \( \text{stdv} \) is the standard deviation of the increment

\( \sqrt{\text{stdv}^2 + \text{stdv}^2} \).
- \( \text{edproc} \) is the standard deviation of the process.

Program parameters:

- \( \text{ndim} \) is the maximum allowable number of analyses
- \( h \) is the grid size for trapezoidal rule numerical integration
- \( lmn \) dimensions local vectors in \( \text{glan} \), NB if this is changed,

user must also change parameters in \( \text{cprob} \), \( \text{fcab} \) and other!!!

Subroutines and functions called directly:

- \text{cprob} computes exit probabilities for current analysis.
- \text{first} computes joint density at first analysis.
- \text{other} computes joint density at second and later analyses.
- \text{error1} checks number of interim analyses.
- \text{error2} checks number of grid points.

\begin{align*}
\text{double precision} & \hspace{1cm} t(*), y_a(*), y_b(*), \text{delta}, \\
& \hspace{1cm} \text{petr}(*), \text{qpos}(*), \text{qneg}(*), \text{est}, \text{pb} \\
\text{integer} & \hspace{1cm} n \\
\text{double precision} & \hspace{1cm} h \\
\text{integer} & \hspace{1cm} \text{ndim}, \text{lmn} \\
\text{parameter} & \hspace{1cm} (\text{ndim}=25, \text{lmn}=5000, h = .05d0)
\end{align*}
\end{verbatim}
double precision ya(ndim),yb(ndim),stdv(ndim),sdproc(ndim),
    last(lmm)
integer nints(ndim)

double precision pnorm

c
Check limit for number of interim analyses (looks).
if(nn.gt.ndim) then
    call errori(nn,ndim)
    return
end if

c
Compute standard deviations of process and increments.
call sd(nn,t,stdv,sdproc)

c
Set upper (yb) and lower (ya) integration limits. Check
limit for number of grid points.
do 10 i=1,nn
    yb(i) = zb(i)**sdproc(i)-delta*t(i)
    ya(i) = za(i)**sdproc(i)-delta*t(i)
    nints(i) = idint((yb(i)-ya(i))/(h*stdv(i))) + 1
    if (nints(i).gt.lmm) then
        call error2(i,lmm)
        return
    end if
10 continue

c
Begin loop calculating probabilities for each look.
c
For first look some direct calculation is possible
using normal cdf. Bounds in standard form are
adjusted for mean of process at t(i).
    pstr(i) = 1.0 - (pnorm(zb(i)-delta*t(i)/stdv(i))
        - pnorm(za(i)-delta*t(i)/stdv(i)))
    qpos(i) = 1.0 - (pnorm(zb(i)-delta*t(i)/stdv(i))
        - pnorm(za(i)-delta*t(i)/stdv(i)))
    qneg(i) =

After the first look, numerical integration is necessary.
do 100 i=2,nn
    Compute density of process at first look.
    if (i.eq.2) then
        call first(ya(i),yb(i),h,stdv(i),last,nints)
    end if

c
Compute pstr, qpos and qneg in subroutine cprob.
call cprob(last,nints,ya,yb,h,stdv(i),
    pstr(i),qpos(i),qneg(i))
    pstr(i) = 1.0 - pstr(i)

c
If density will be needed for next step, call other.
if (i.ne.nn) then
    call other(ya,yb,h,stdv(i),h, last,nints)
end if

c
Accumulate boundary crossing probability. (Not implemented.)
    pb = pb + (qpos(i)+qneg(i))
    est = est + (qpos(i)+qneg(i))*t(i)
100 continue

c
Compute expected stopping time. (Not implemented.)
    est = 1.0 - est

return
end

subroutine sd(nn,t, sdincr,sdproc)

c
Standard deviations are calculated. Users may
modify for particular applications, eg rescaling
by stdv(i)/stdv(nn):
    sdincr(i) = dsqrt(t(i)/t(nn))
    sdproc(i) = dsqrt(t(i)/t(nn))

39
double precision t(*), sdincr(*), sdproc(*)
integer na

integer i

Loop for standard deviations
do 100 i=1,na

if (i.eq.1) then
  sdincr(i) = dsqrt(t(i))
  sdproc(i) = sdincr(i)
else
  Standard deviation for second and later analyses:
  sdincr(i) = dsqrt((t(i)-t(i-1)))
  sdproc(i) = dsqrt(t(i))
end if
100 continue

return
end

subroutine cprob(last,nints,ya,yb,i,h,stdv,pstr,qpos,qneg)

Compute probabilities for current analysis. Applying
Fubini's Theorem to Armitage, McPherson and Rowe's
formulae, the inner integral is a normal cdf and the outer
is computed by a trapezoidal rule numerical integration.

Input variables:
  last is the vector of density values from the previous step.
  ya is the vector of lower integration limits.
  yb is the vector of upper integration limits.
  i is the index of the current step.
  h is the grid size.
  stdv is the standard deviation of the process increment.

Returned variables:
  pstr is the value of the integral from ya to yb.
  qpos is the value of the integral from yb to infinity.
  qneg is the value of the integral from ya to minus infinity.

Local variables:
  pmid is vector of integrand values for pstr.
  pupr is vector of integrand values for qpos.
  plow is vector of integrand values for qneg.
  grid is the argument for pmid, pupr and plow.
  nints is the number of steps of size h between ya and yb.
  hlast is the grid size from last step.

Program parameters:
  nwork dimensions pmid, pupr, plow.

Subroutines and functions called directly:
  pnorm is the standard normal cdf.
  trap is a trapezoidal rule numerical integration routine.

double precision last(*), ya(*), yb(*), h, stdv, pstr, qpos, qneg
integer i,nints(*)

integer nwork
parameter(nwork=5000)

double precision pmid(nwork), pupr(nwork), plow(nwork),
  grid, tpstr, tpqos, tqneg, hlast
integer j

double precision pnorm

Previous grid size.
  hlast = (yb(i-1)-ya(i-1))/nints(i-1)

Function values to be passed to numerical integration
routine are calculated.
do 100 j=1,nints(i-1)+1
  grid = ya(i-1) + (hlast*(j-1))
  pmid(j) = pnorm((yb(i)-grid)/stdv) - pnorm((ya(i)-grid)/stdv) * last(j)
F Routines common to both programs

subroutine first(ya,yb,h,stdv,last,nints)

Joint density at first step is the normal.

Input variables:
y_a is the lower integration limit for the first analysis.
y_b is the upper integration limit for the first analysis.
h is the grid size determining arguments for function values.
stdv is the standard deviation of the process at first analysis.

Returned variables:
last is the vector of function values over the grid.
nints is the number of whole steps of size h between ya and y_b.

Local variables:
h is the grid size used.

Subroutines and functions called directly:
g is the standard normal density function.

double precision ya,yb,last(*),stdv,h
integer nints(*)

double precision hh
integer j
double precision g

Compute grid size to be used.
hh = (y_b - y_a)/nints(1)

Evaluate function (normal density) at grid points.
do 10 j = 1,nints(1)+1
   last(j) = g((ya+(hh*(j-1)))/stdv)/stdv
10 continue

return
end

subroutine other(ya,yb,i,stdv,h,last,nints)

Computes joint density for current step using fcab.

Input variables:
y_a is the vector of lower integration limits.
y_b is the vector of upper integration limits.
h is the grid size determining arguments for function values.
stdv is the standard deviation of the process increment.
last on entry is the density values from the previous step.

Returned variables:
last is the vector of function values over the grid.
nints is the number of whole steps of size h between ya and y_b.

Local variables:
ggrid is the argument for function evaluation.
fn is a temporary vector of function values—last from
previous step is needed to compute each current value.
hh is the actual grid size used.

Program parameters:
nlim dimensions fn. Must be compatible with dimension of last.
Subroutines and functions called directly:
f cab computes current values of the joint density.

double precision ya(*),yb(*),stdv,h,last(*)
integer i,nints(*)

integer nlim
parameter(nlim=5000)

double precision grid,fn(nlim),hh,h,last
integer j

double precision f cab

Current grid size to be used.
hh = (yb(i+1)-ya(i))/nints(i)

Previous grid size.
hlast = (yb(i+1)-ya(i))/nints(i)

evaluate function over grid ya+(j-1)*hh, j=1,nints+1

do 10 j=1,nints+1
   grid = ya(i)+(hh*(j-1))
10 vec(j) = f cab(last,nints(i-1),ya(i-1),yb(i-1),hlast,grid,stdv)

continue

Copy fn into last.
do 20 j=1,nints+1
   last(j) = fn(j)
20 continue

return
double precision function f cab(last,nints,yaml,ybmi,h,x,stdv)

This function computes the density function of (W(t(i)),t(i))
where t(i) are the stopping times and W(t(i)) is the standard
Brownian motion process on (0,1).

The joint density for the previous step is multiplied by
the density for the increment to the current step (just the
normal density with proper mean and variance), and the
result is numerically integrated.

Input variables:
last is the vector of values of the previous density.
nints is the number of steps of size h between yaml and ybmi.
yaml is the lower integration limit for the previous step.
ybmi is the upper integration limit for the previous step.
h is the grid size.
x is the argument at which density function is evaluated.
stdv is the square root of the diffusion of a Brownian motion.

Local variables:
f is a temporary vector of function values to pass to f trap.

Program parameters:
nlim dimensions fn. Must be compatible with dimension of last.
Subroutines and functions called directly:
g is the standard normal density function.
trap is a trapezoidal rule numerical integration.

double precision last(*),yaml,ybmi,h,x,stdv

integer nints

integer nlim
parameter(nlim=5000)
double precision f(xlim), grid, tfcab
integer j
double precision g

evaluate function over grid
do 10 j=1,nints+1
   grid = yaml + (h*(j-1))
   f(j) = last(j) + (g((x-grid)/stdv)/stdv)
10 continue

numerical integration.
call trap(f,nints,yaml,ybml,h, tfcab)
tf cab = tf cab
return
end

subroutine trap(f,n,a,b,h, area)
simplest trapezoidal rule numerical integration.

input variables:
f is a vector of function values.
n is the number of trapezoids with base h.
a is the lower integration limit, the first grid point.
b is the upper integration limit, the last grid point.
h is the grid size.

returned variables:
area is the sum of the specified function values.

local variables:
sum accumulates the area of the trapezoids.

double precision f(+), a, b, h, area
integer n

double precision sum

initialize sum with the first function value.
sum = f(1)

all but the first and last function values appear twice in the summation.
do 100 j=2, n
   sum = sum + 2.0*f(j)
100 continue

add the last function value.
sum = sum + f(n+1)

multiply the sum by h/2.
area = (h/2.0)*sum

return
end

double precision function g(x)
g(x) is the normal (0,1) density function.
double precision x, pi
parameter(pi=3.1415926535864)
g = dexp(-0.5d0*x*x)/dsqrt(2.d0*pi)
return
end

double precision function znorm(p)

znorm is the inverse normal cdf.
from statlib@temper.stat.cmu.edu ("send 241 from apstat") with
only minor modification.

algorithm as241 appl. statist. (1998) vol. 37, no. 3

produces the normal deviate z corresponding to a given lower
tail area of p; z is accurate to about 1 part in 10**16.
The hash sums below are the sums of the mantissas of the coefficients. They are included for use in checking transcription.

DOUBLE PRECISION ZERO, ONE, HALF, SPLIT1, SPLIT2, CONST1, CONST2, A0, A1, A2, A3, A4, A5, A6, A7, B1, B2, B3, C0, C1, C2, C3, C4, C5, C6, C7, D1, D2, D3, D4, D5, D6, D7, E0, E1, E2, E3, E4, E5, E6, E7, F1, F2, F3, F4, F5, F6, F7, F, Q, R
PARAMETER (ZERO = 0.0D0, ONE = 1.0D0, HALF = 0.5D0, SPLIT1 = 0.425D0, SPLIT2 = 5.0D0, CONST1 = 0.186286D0, CONST2 = 1.0D0)

Coefficients for P close to 0.5

HASH SUM AB 55.66319 26006 14901 4439

Coefficients for P not close to 0, 0.5 or 1.

PARAMETER (C0 = 1.42343 71107 49683 57734D0, C1 = 4.63033 78481 56545 29590D0, C2 = 6.76949 72214 60691 4085D0, C3 = 3.46784 82347 03204 0504D0, C4 = 1.27045 82624 52368 3825D0, C5 = 2.41780 72517 74306 1177D0-1, C6 = 2.27238 44989 26918 4583D-2, C7 = 7.74545 01427 83414 0764D-4, D1 = 2.05319 16266 73758 8218D0, D2 = 1.67638 46301 83903 8490D0, D3 = 6.89767 33496 51000 0465D-1, D4 = 1.48103 97642 74800 7490D-1, D5 = 1.51986 66503 61645 7166D-2, D6 = 8.47593 80849 95344 9600D-4, D7 = 1.05075 00716 44416 84224D-9)
HASH SUM CD 49.33266 50330 16102 89036

Coefficients for P near 0 or 1.

PARAMETER (E0 = 6.65790 46435 01103 7772D0, E1 = 5.46378 49111 64114 3699D0, E2 = 1.78482 65399 17291 3350D0, E3 = 2.96560 57182 85048 9123D0-1, E4 = 2.65321 89526 57612 3093D0-2, E5 = 1.24266 09473 88078 4366D0-3, E6 = 7.11555 55687 43487 5815D-5, E7 = 2.01033 43992 92268 1326D-7, F1 = 5.99832 20655 58879 3799D0-1, F2 = 1.36929 88092 72358 0510D-1, F3 = 1.48763 61290 85061 4952D-2, F4 = 7.86669 13114 56132 5910D0-4, F5 = 1.84631 83175 10054 6818D-5, F6 = 1.42161 17683 16445 8887D-7, F7 = 2.04426 31033 89939 7864D-15)
HASH SUM EF 47.52583 31754 92896 71629

IFault = 0
Q = P - HALF
IF (ABS(Q) LE SPLIT1) THEN
R = CST1 - Q * Q
*                  * R + A2) * R + A1) * R + A0) / 
*                  * ((((((B7 * R + B6) * R + B5) * R + B4) * R + B3) 
*                  * * R + B2) * R + B1) * R + B0) 
RETURN                
ELSE                
IF (Q .LT. ZERO) THEN 
R = P 
ELSE                
R = ONE - P 
END IF                
IF (R .LE. ZERO) THEN 
IFault = 1 
znorm = ZERO 
RETURN                
END IF                
R = SQRT(-LOG(R))                
IF (R .LE. SPLT2) THEN 
R = R - CONST2 
znorm = ((((((C7 * R + C6) * R + C5) * R + C4) * R + C3) 
*                  * * R + C2) * R + C1) * R + C0) / 
*                  * ((((((D7 * R + D6) * R + D5) * R + D4) * R + D3) 
*                  * * R + D2) * R + D1) * R + ONE) 
ELSE                
R = R - SPLT2 
*                  * * R + E2) * R + E1) * R + E0) / 
*                  * ((((((F7 * R + F6) * R + F5) * R + F4) * R + F3) 
*                  * * R + F2) * R + F1) * R + ONE) 
END IF                
IF (Q .LT. ZERO) znorm = - znorm 
RETURN                
END IF                
                    
c double precision function pnorm(Z)         
c pnorm is the normal cdf. 
From statlib@temper.stat.cmu.edu ("send 66 from apstat") with 
changes: subroutine to function. 
c REFERENCE: ADAMS, A.G. AREAS UNDER THE NORMAL CURVE, 
c P, Q = PROBABILITIES TO THE LEFT AND RIGHT OF Z 
c FOR THE STANDARD NORMAL DISTRIBUTION. 
c PDF = THE PROBABILITY DENSITY FUNCTION 
c LATEST REVISION - 23 JANUARY 1961 
c ********************************************************************************** 
c IMPLICIT DOUBLE PRECISION (A-H,O-Z) 
DATA A0,A1,A2,A3,A4,A5,A6,A7/0.8969422804440000, 
          1 0.3999034385048000, 5.7688546046800000, 29.82138578060000, 
          2 2.0243312167990000, 48.66993069200000, 5.9288774438000000, 
          3 0.9957329325302000, 1.00000015302000, 3.0994226088000000, 
          4 3.06220001000000, 1.00000015302000, 3.0994226088000000, 1.00000015302000, 
          5 0.9516791163500000, 5.2933032492000000, 4.8385912808000000, 
          6 15.1560972481000000, 0.7423809240270000, 30.78993303400000, 
          7 3.9901941470100000/ 
C ZABS = ABS(Z) 
IF(ZABS.GT.12.7DO) GO TO 20 
I = A0*Z*Z 
PDF = EXP(-Y)*BO 
IF(ZABS.GT.1.28DO) GO TO 10 
C C Z BETWEEN -1.28 AND +1.28 
C Q = A0-ZABS*(A1-A2*Y/(Y+A3-A4)/(Y+A6+A6/(Y+A7))) 
IF(Z.LT.0.D0) GO TO 30 
pnorm = 1.DO-Q 
RETURN 
C C ZABS BETWEEN 1.28 AND 12.7
C 10  Q = PDF/(ZABS-B1+B2/(ZABS+B3+B4)/(ZABS-B5+B6)/(ZABS+B7-B8/1
   (ZABS+B9+B10/(ZABS+B11))))
   IF(Z.LT.0.D0) GO TO 30
   pnorm = 1.D0-Q
   RETURN

C C Z FAR OUT IN TAIL
C
C 20  Q = 0.D0
   PDF = 0.D0
   IF(Z.LT.0.D0) GO TO 30
   pnorm = 1.D0
   RETURN

C C NEGATIVE Z, INTERCHANGE P AND Q
C
C 30  pnorm = Q
   Q = 1.D0-P
   RETURN
   END