

NAG Toolbox

nag_sum_invlaplace_crump (c06la)

1 Purpose

`nag_sum_invlaplace_crump (c06la)` estimates values of the inverse Laplace transform of a given function using a Fourier series approximation. Real and imaginary parts of the function, and a bound on the exponential order of the inverse, are required.

2 Syntax

```
[valinv, errest, nterms, na, alow, ahigh, nfeval, ifail] =
nag_sum_invlaplace_crump(fun, t, relerr, alphab, 'n', n, 'tfac', tfac, 'mxterm',
mxterm)

[valinv, errest, nterms, na, alow, ahigh, nfeval, ifail] = c06la(fun, t,
relerr, alphab, 'n', n, 'tfac', tfac, 'mxterm', mxterm)
```

3 Description

Given a function $F(p)$ defined for complex values of p , `nag_sum_invlaplace_crump (c06la)` estimates values of its inverse Laplace transform by Crump's method (see Crump (1976)). (For a definition of the Laplace transform and its inverse, see the C06 Chapter Introduction.)

Crump's method applies the epsilon algorithm (see Wynn (1956)) to the summation in Durbin's Fourier series approximation (see Durbin (1974))

$$f(t_j) \simeq \frac{e^{at_j}}{\tau} \left[\frac{1}{2} F(a) - \sum_{k=1}^{\infty} \left\{ \operatorname{Re} \left(F \left(a + \frac{k\pi i}{\tau} \right) \right) \cos \frac{k\pi t_j}{\tau} - \operatorname{Im} \left(F \left(a + \frac{k\pi i}{\tau} \right) \right) \sin \frac{k\pi t_j}{\tau} \right\} \right],$$

for $j = 1, 2, \dots, n$, by choosing a such that a prescribed relative error should be achieved. The method is modified slightly if $t = 0.0$ so that an estimate of $f(0.0)$ can be obtained when it has a finite value. τ is calculated as $t_{\text{fac}} \times \max(0.01, t_j)$, where $t_{\text{fac}} > 0.5$. You specify t_{fac} and α_b , an upper bound on the exponential order α of the inverse function $f(t)$. α has two alternative interpretations:

- (i) α is the smallest number such that

$$|f(t)| \leq m \times \exp(\alpha t)$$

for large t ,

- (ii) α is the real part of the singularity of $F(p)$ with largest real part.

The method depends critically on the value of α . See Section 9 for further details. The function calculates at least two different values of the argument a , such that $a > \alpha_b$, in an attempt to achieve the requested relative error and provide error estimates. The values of t_j , for $j = 1, 2, \dots, n$, must be supplied in monotonically increasing order. The function calculates the values of the inverse function $f(t_j)$ in decreasing order of j .

4 References

Crump K S (1976) Numerical inversion of Laplace transforms using a Fourier series approximation *J. Assoc. Comput. Mach.* **23** 89–96

Durbin F (1974) Numerical inversion of Laplace transforms: An efficient improvement to Dubner and Abate's method *Comput. J.* **17** 371–376

Wynn P (1956) On a device for computing the $e_m(S_n)$ transformation *Math. Tables Aids Comput.* **10** 91–96

5 Parameters

5.1 Compulsory Input Parameters

1: **fun** – SUBROUTINE, supplied by the user.

fun must evaluate the real and imaginary parts of the function $F(p)$ for a given value of p .

```
[fr, fi] = fun(pr, pi)
```

Input Parameters

- 1: **pr** – REAL (KIND=nag_wp)
 2: **pi** – REAL (KIND=nag_wp)

The real and imaginary parts of the argument p .

Output Parameters

- 1: **fr** – REAL (KIND=nag_wp)
 2: **fi** – REAL (KIND=nag_wp)

The real and imaginary parts of the value $F(p)$.

2: **t(n)** – REAL (KIND=nag_wp) array

Each $\mathbf{t}(j)$ must specify a point at which the inverse Laplace transform is required , for $j = 1, 2, \dots, n$.

Constraint: $0.0 \leq \mathbf{t}(1) < \mathbf{t}(2) < \dots < \mathbf{t}(n)$.

3: **relerr** – REAL (KIND=nag_wp)

The required relative error in the values of the inverse Laplace transform. If the absolute value of the inverse is less than **relerr**, then absolute accuracy is used instead. **relerr** must be in the range $0.0 \leq \mathbf{relerr} < 1.0$. If **relerr** is set too small or to 0.0, then the function uses a value sufficiently larger than **machine precision**.

4: **alphab** – REAL (KIND=nag_wp)

α_b , an upper bound for α (see Section 3). Usually, α_b should be specified equal to, or slightly larger than, the value of α . If $\alpha_b < \alpha$ then the prescribed accuracy may not be achieved or completely incorrect results may be obtained. If α_b is too large nag_sum_invlaplace_crump (c06la) will be inefficient and convergence may not be achieved.

Note: it is as important to specify α_b correctly as it is to specify the correct function for inversion.

5.2 Optional Input Parameters

1: **n** – INTEGER

Default: the dimension of the array **t**.

n , the number of points at which the value of the inverse Laplace transform is required.

Constraint: $\mathbf{n} \geq 1$.

2: **tfac** – REAL (KIND=nag_wp)

Suggested value: **tfac** = 0.8.

Default: 0.8

t_{fac} , a factor to be used in calculating the parameter τ . Larger values (e.g., 5.0) may be specified for difficult problems, but these may require very large values of **mxterm**.

Constraint: **tfac** > 0.5.

3: **mxterm** – INTEGER

Suggested value: **mxterm** ≥ 100, except for very simple problems.

Default: 100

The maximum number of (complex) terms to be used in the evaluation of the Fourier series.

Constraint: **mxterm** ≥ 1.

5.3 Output Parameters

1: **valinv(n)** – REAL (KIND=nag_wp) array

An estimate of the value of the inverse Laplace transform at $t = \mathbf{t}(j)$, for $j = 1, 2, \dots, n$.

2: **errest(n)** – REAL (KIND=nag_wp) array

An estimate of the error in **valinv(j)**. This is usually an estimate of relative error but, if **valinv(j)** < **relerr**, **errest(j)** estimates the absolute error. **errest(j)** is unreliable when **valinv(j)** is small but slightly greater than **relerr**.

3: **nterms** – INTEGER

The number of (complex) terms actually used.

4: **na** – INTEGER

The number of values of a used by the function. See Section 9.

5: **alow** – REAL (KIND=nag_wp)

The smallest value of a used in the algorithm. This may be used for checking the value of **alphab** – see Section 9.

6: **ahigh** – REAL (KIND=nag_wp)

The largest value of a used in the algorithm. This may be used for checking the value of **alphab** – see Section 9.

7: **nfeval** – INTEGER

The number of calls to **fun** made by the function.

8: **ifail** – INTEGER

ifail = 0 unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Note: nag_sum_invlaplace_crump (c06la) may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the function:

ifail = 1

On entry, **n** < 1,
or **mxterm** < 1,
or **relerr** < 0.0,

or $\text{relerr} \geq 1.0$,
 or $\text{tfac} \leq 0.5$.

ifail = 2

On entry, $t(1) < 0.0$,
 or $t(1), t(2), \dots, t(n)$ are not in strictly increasing order.

ifail = 3

$t(n)$ is too large for this value of **alphab**. If necessary, scale the problem as described in Section 9.

ifail = 4

The required accuracy cannot be obtained. It is possible that **alphab** is less than α . Alternatively, the problem may be especially difficult. Try increasing **tfac**, **alphab** or both.

ifail = 5

Convergence failure in the epsilon algorithm. Some values of **valinv(j)** may be calculated to the desired accuracy; this may be determined by examining the values of **errest(j)**. Try reducing the range of **t** or increasing **mxterm**. If **ifail = 5** still results, try reducing **tfac**.

ifail = 6 (warning)

All values of **valinv(j)** have been calculated but not all are to the requested accuracy; the values of **errest(j)** should be examined carefully. Try reducing the range of **t**, or increasing **tfac**, **alphab** or both.

ifail = -99

An unexpected error has been triggered by this routine. Please contact NAG.

ifail = -399

Your licence key may have expired or may not have been installed correctly.

ifail = -999

Dynamic memory allocation failed.

7 Accuracy

The error estimates are often very close to the true error but, because the error control depends on an asymptotic formula, the required error may not always be met. There are two principal causes of this: Gibbs' phenomena, and zero or small values of the inverse Laplace transform.

Gibbs' phenomena (see the C06 Chapter Introduction) are exhibited near $t = 0.0$ (due to the method) and around discontinuities in the inverse Laplace transform $f(t)$. If there is a discontinuity at $t = c$ then the method converges such that $f(c) \rightarrow (f(c-) + f(c+))/2$.

Apparent loss of accuracy, when $f(t)$ is small, may not be serious. Crump's method keeps control of relative error so that good approximations to small function values may appear to be very inaccurate. If $|f(t)|$ is estimated to be less than **relerr** then this function switches to absolute error estimation. However, when $|f(t)|$ is slightly larger than **relerr** the relative error estimates are likely to cause **ifail = 6**. If this is found inconvenient it can sometimes be avoided by adding k/p to the function $F(p)$, which shifts the inverse to $k + f(t)$.

Loss of accuracy may also occur for highly oscillatory functions.

More serious loss of accuracy can occur if α is unknown and is incorrectly estimated. See Section 9.

8 Further Comments

8.1 Timing

The value of n is less important in general than the value of **nterms**. Unless **fun** is very inexpensive to compute, the timing is proportional to **na** × **nterms**. For simple problems **na** = 2 but in difficult problems **na** may be somewhat larger.

8.2 Precautions

You are referred to the C06 Chapter Introduction for advice on simplifying problems with particular difficulties, e.g., where the inverse is known to be a step function.

The method does not work well for large values of t when α is positive. It is advisable, especially if **ifail** = 3 is obtained, to scale the problem if $|\alpha|$ is much greater than 1.0. See the C06 Chapter Introduction.

The range of values of t specified for a particular call should not be greater than about 10 units. This is because the method uses arguments based on the value **t(n)** and these tend to be less appropriate as t becomes smaller. However, as the timing of the function is not especially dependent on n , it is usually far more efficient to evaluate the inverse for ranges of t than to make separate calls to the function for each value of t .

The most important argument to specify correctly is **alphab**, an upper bound for α . If, on entry, **alphab** is sufficiently smaller than α then completely incorrect results will be obtained with **ifail** = 0. Unless α is known theoretically it is strongly advised that you should test any estimated value used. This may be done by specifying a single value of t (i.e **t(n)**, $n = 1$) with two sets of suitable values of **tfac**, **relerr** and **mxterm**, and examining the resulting values of **alow** and **ahigh**. The value of **t(1)** should be chosen very carefully and the following points should be borne in mind:

- (i) **t(1)** should be small but not too close to 0.0 because of Gibbs' phenomenon (see Section 7),
- (ii) the larger the value of **t(1)**, the smaller the range of values of a that will be used in the algorithm,
- (iii) **t(1)** should ideally not be chosen such that $f(\mathbf{t}(1)) = 0.0$ or a very small value. For suitable problems **t(1)** might be chosen as, say, 0.1 or 1.0 depending on these factors. The function calculates **alow** from the formula

$$\mathbf{alow} = \mathbf{alphab} - \frac{\ln(0.1 \times \mathbf{relerr})}{2 \times \tau}.$$

Additional values of a are computed by adding $1/\tau$ to the previous value. As $\tau = \mathbf{tfac} \times \mathbf{t}(n)$, it will be seen that large values of **tfac** and **relerr** will test for a close to **alphab**. Small values of **tfac** and **relerr** will test for a large. If the result of both tests is **ifail** = 0, with comparable values for the inverse, then this gives some credibility to the chosen value of **alphab**. You should note that this test could be more computationally expensive than the calculation of the inverse itself. The example program (see Section 10) illustrates how such a test may be performed.

9 Example

This example estimates the inverse Laplace transform of the function $F(p) = 1/(p + 1/2)$. The true inverse of $F(p)$ is $\exp(-t/2)$. Two preliminary calls to the function are made to verify that the chosen value of **alphab** is suitable. For these tests the single value **t(1)** = 1.0 is used. To test values of a close to **alphab**, the values **tfac** = 5.0 and **relerr** = 0.01 are chosen. To test larger a , the values **tfac** = 0.8 and **relerr** = 1.0e–3 are used. Because the values of the computed inverse are similar and **ifail** = 0 in each case, these tests show that there is unlikely to be a singularity of $F(p)$ in the region $-0.04 \leq \text{Re}(p) \leq 6.51$.

9.1 Program Text

```

function c06la_example

fprintf('c06la example results\n\n');

% Initialize variables and arrays.
mxterm = nag_int(200);
alphab = -0.5;
n = 1;
trures = zeros(1, n);
trurel = zeros(1, n);

% First, make two preliminary calls to the routine to verify that the
% chosen alphab value is suitable. Use t = 1 for these calls.
t(1) = 1;
disp(['Test with t(1) = ',num2str(t(1))]);

% Parameter values for a close to alphab.
tfac = 7.5;
relerr = 1e-2;
fprintf(['\nmxterm = %3.0f    tfac = %3.2f    alphab = %3.2f ', ...
    'relerr = %3.0e \n\n'], mxterm, tfac, alphab, relerr);

[valinv, errest, nterms, na, alow, ahigh, nfeval, ifail] = ...
c06la( ...
    @fun, t, relerr, alphab, 'tfac', tfac, 'mxterm', mxterm);

% Calculate results and output them.
trures(1) = exp(double(-t(1)/2));
trurel(1) = abs((valinv(1)-trures(1))/trures(1));
disp('t      Result      exp(t/2)      Relative error      Error estimate')
fprintf('%1.1f    %8.4f    %8.4f    %8.4f    %8.4f\n', ...
    t(1), valinv(1), trures(1), trurel(1), errest(1));
fprintf(['\nnterms = %3d    nfeval = %3d    alow = %3.2f ', ...
    'ahigh = %3.2f    ifail = %3d\n\n'], nterms, nfeval, alow, ...
    ahigh, ifail);

% Parameter values for larger a.
tfac = 0.8;
relerr = 1e-3;
disp(['Test with t(1) = ',num2str(t(1))]);
fprintf(['\nmxterm = %3d    tfac = %3.2f    alphab = %3.2f ', ...
    'relerr = %3.0e \n\n'], mxterm, tfac, alphab, relerr);

[valinv, errest, nterms, na, alow, ahigh, nfeval, ifail] = ...
c06la( ...
    @fun, t, relerr, alphab, 'tfac', tfac, 'mxterm', mxterm);

% Calculate results and output them.
trures(1) = exp(double(-t(1)/2));
trurel(1) = abs((valinv(1)-trures(1))/trures(1));
disp('t      Result      exp(t/2)      Relative error      Error estimate')
fprintf('%1.1f    %8.4f    %8.4f    %8.5f    %8.5f\n', ...
    t(1), valinv(1), trures(1), trurel(1), errest(1));
fprintf(['\nnterms = %3d    nfeval = %3d    alow = %3.2f ', ...
    'ahigh = %3.2f    ifail = %3d\n\n'], nterms, nfeval, alow, ...
    ahigh, ifail);

% Now calculate the inverse Laplace transform for several t values.
n = 5;
t = [1:5];
disp('Compute inverse')
fprintf(['\nmxterm = %3d    tfac = %3.2f    alphab = %3.2f ', ...
    'relerr = %3.0e \n\n'], mxterm, tfac, alphab, relerr);

[valinv, errest, nterms, na, alow, ahigh, nfeval, ifail] = ...
c06la( ...
    @fun, t, relerr, alphab, 'tfac', tfac, 'mxterm', mxterm);

% Calculate results and output them.

```

```

disp('t      Result      exp(t/2)    Relative error    Error estimate')
for i = 1:n
    trures(i) = exp(-t(i)/2);
    trurel(i) = abs((valinv(i)-trures(i))/trures(i));
    fprintf('%1.0f    ',t(i));
    fprintf(' %4.3f    %4.3f    %4.3e    %4.3e\n',...
            valinv(i),trures(i),trurel(i),errest(i));
end
fprintf(['\nmxterm = %3d    tfac = %3.2f    alphab = %3.2f ', ...
         'relerr = %3.0e \n\n'], mxterm, tfac, alphab, relerr);

% Plot results.
fig1 = figure;
display_plot(t,trures,valinv,trurel,errest);

function [fr,fi] = fun(preal,pimag)
    % Evaluate the real & imaginary parts of the user function.
    z = complex(1)/complex(preal+0.5,pimag);
    fr = real(z);
    fi = imag(z);

function display_plot(t, trures, valinv, trurel, errest)
    % Use a log plot for the second curve.
    [haxes, hline1, hline2] = plotyy(t, valinv, t, trurel, 'plot', 'semilogy');
    % Add the third curve (as points).
    hold on
    hpoints = plot(t, trures, '+', 'MarkerEdgeColor', 'r');
    % Set the axis limits and the tick specifications to beautify the plot.
    set(haxes(1), 'YLim', [0 0.8]);
    set(haxes(1), 'YMinorTick', 'on');
    set(haxes(1), 'YTick', [0.0 0.2 0.4 0.6 0.8]);
    set(haxes(2), 'YLim', [1e-7 5e-3]);
    set(haxes(2), 'YMinorTick', 'on');
    set(haxes(2), 'YTick', [1e-7 1e-6 1e-5 1e-4]);
    for iaxis = 1:2
        % These properties must be the same for both sets of axes.
        set(haxes(iaxis), 'XLim', [1 5]);
        set(haxes(iaxis), 'XTick', [1 1.5 2 2.5 3 3.5 4 4.5 5]);
    end
    set(gca, 'box', 'off'); % no ticks on opposite axes.
    % Add title.
    title('Inverse Laplace Transform of 1/(p+0.5)');
    % Label the x axis.
    xlabel('t');
    % Label the left and right y axes.
    ylabel(haxes(1),'f(t)');
    set(haxes(2),'YLim',[1e-8 Inf]);
    ylabel(haxes(2),'Error');
    % Add the fourth curve (plotted against the log axis).
    axes(haxes(2));
    hold on;
    hline3 = semilogy(t, errest);
    % Set some features of the three lines.
    set(hline1, 'LineWidth', 0.5, 'MarkerFaceColor', 'auto');
    set(hline2, 'LineWidth', 0.5, 'Marker', 'x', 'Color','Magenta');
    set(hline3, 'LineWidth', 0.5, 'Marker', '*', 'MarkerFaceColor', 'auto');
    % Add legend.
    legend([hpoints hline1 hline2 hline3], 'result', 'exp(-t/2)', ...
           'relative error', 'estimated error', 'Location','SouthWest');

```

9.2 Program Results

c06la example results

Test with $t(1) = 1$

```

mxterm = 200    tfac = 7.50    alphab = -0.50 relerr = 1e-02
t      Result      exp(t/2)    Relative error    Error estimate
1.0      0.6071      0.6065      0.0010      0.0037

```

```

nterms = 18    nfeval = 36    alow = -0.04 ahigh = 0.09    ifail = 0
Test with t(1) = 1
mxterm = 200    tfac = 0.80    alphab = -0.50 relerr = 1e-03
t      Result      exp(t/2)  Relative error  Error estimate
1.0    0.6065     0.6065    0.00002     0.00008
nterms = 13    nfeval = 28    alow = 5.26 ahigh = 6.51    ifail = 0
Compute inverse
mxterm = 200    tfac = 0.80    alphab = -0.50 relerr = 1e-03
t      Result      exp(t/2)  Relative error  Error estimate
1    0.607      0.607     4.746e-05   3.155e-04
2    0.368      0.368     6.910e-06   9.386e-05
3    0.223      0.223     1.589e-05   7.839e-05
4    0.135      0.135     1.353e-05   7.504e-05
5    0.082      0.082     2.014e-05   8.080e-05
mxterm = 200    tfac = 0.80    alphab = -0.50 relerr = 1e-03

```

