CHAPTER

THIRTYTHREE

SERIES ACCELERATION

The functions described in this chapter accelerate the convergence of a series using the Levin u-transform. This method takes a small number of terms from the start of a series and uses a systematic approximation to compute an extrapolated value and an estimate of its error. The u-transform works for both convergent and divergent series, including asymptotic series.

These functions are declared in the header file `gsl_sum.h`.

33.1 Acceleration functions

The following functions compute the full Levin u-transform of a series with its error estimate. The error estimate is computed by propagating rounding errors from each term through to the final extrapolation.

These functions are intended for summing analytic series where each term is known to high accuracy, and the rounding errors are assumed to originate from finite precision. They are taken to be relative errors of order `GSL_DBL_EPSILON` for each term.

The calculation of the error in the extrapolated value is an $O(N^2)$ process, which is expensive in time and memory. A faster but less reliable method which estimates the error from the convergence of the extrapolated value is described in the next section. For the method described here a full table of intermediate values and derivatives through to $O(N)$ must be computed and stored, but this does give a reliable error estimate.

gsl_sum_levin_u_workspace
Workspace for a Leven u-transform.

gsl_sum_levin_u_workspace * **gsl_sum_levin_u_alloc** (size_t *n*)
This function allocates a workspace for a Levin u-transform of `n` terms. The size of the workspace is $O(2n^2 + 3n)$.

void **gsl_sum_levin_u_free** (*gsl_sum_levin_u_workspace* * *w*)
This function frees the memory associated with the workspace `w`.

int **gsl_sum_levin_u_accel** (const double * *array*, size_t *array_size*, *gsl_sum_levin_u_workspace* * *w*, double * *sum_accel*, double * *abserr*)
This function takes the terms of a series in `array` of size `array_size` and computes the extrapolated limit of the series using a Levin u-transform. Additional working space must be provided in `w`. The extrapolated sum is stored in `sum_accel`, with an estimate of the absolute error stored in `abserr`. The actual term-by-term sum is returned in `w->sum_plain`. The algorithm calculates the truncation error (the difference between two successive extrapolations) and round-off error (propagated from the individual terms) to choose an optimal number of terms for the extrapolation. All the terms of the series passed in through `array` should be non-zero.

33.2 Acceleration functions without error estimation

The functions described in this section compute the Levin u-transform of series and attempt to estimate the error from the "truncation error" in the extrapolation, the difference between the final two approximations. Using this method avoids the need to compute an intermediate table of derivatives because the error is estimated from the behavior of the extrapolated value itself. Consequently this algorithm is an $O(N)$ process and only requires $O(N)$ terms of storage. If the series converges sufficiently fast then this procedure can be acceptable. It is appropriate to use this method when there is a need to compute many extrapolations of series with similar convergence properties at high-speed. For example, when numerically integrating a function defined by a parameterized series where the parameter varies only slightly. A reliable error estimate should be computed first using the full algorithm described above in order to verify the consistency of the results.

gsl_sum_levin_utrunc_workspace
Workspace for a Levin u-transform without error estimation

gsl_sum_levin_utrunc_workspace * **gsl_sum_levin_utrunc_alloc** (size_t *n*)
This function allocates a workspace for a Levin u-transform of `n` terms, without error estimation. The size of the workspace is $O(3n)$.

void **gsl_sum_levin_utrunc_free** (*gsl_sum_levin_utrunc_workspace* * *w*)
This function frees the memory associated with the workspace `w`.

int **gsl_sum_levin_utrunc_accel** (const double * *array*, size_t *array_size*, *gsl_sum_levin_utrunc_workspace* * *w*, double * *sum_accel*, double * *abserr_trunc*)
This function takes the terms of a series in `array` of size `array_size` and computes the extrapolated limit of the series using a Levin u-transform. Additional working space must be provided in `w`. The extrapolated sum is stored in `sum_accel`. The actual term-by-term sum is returned in `w->sum_plain`. The algorithm terminates when the difference between two successive extrapolations reaches a minimum or is sufficiently small. The difference between these two values is used as estimate of the error and is stored in `abserr_trunc`. To improve the reliability of the algorithm the extrapolated values are replaced by moving averages when calculating the truncation error, smoothing out any fluctuations.

33.3 Examples

The following code calculates an estimate of $\zeta(2) = \pi^2/6$ using the series,

$$\zeta(2) = 1 + 1/2^2 + 1/3^2 + 1/4^2 + \ldots$$

After `N` terms the error in the sum is $O(1/N)$, making direct summation of the series converge slowly.

```
#include <stdio.h>
#include <gsl/gsl_math.h>
#include <gsl/gsl_sum.h>

#define N 20

int
main (void)
{
  double t[N];
  double sum_accel, err;
  double sum = 0;
  int n;
```

(continues on next page)

(continued from previous page)

```
  gsl_sum_levin_u_workspace * w
    = gsl_sum_levin_u_alloc (N);

  const double zeta_2 = M_PI * M_PI / 6.0;

  /* terms for zeta(2) = \sum_{n=1}^{\infty} 1/n^2 */

  for (n = 0; n < N; n++)
    {
      double np1 = n + 1.0;
      t[n] = 1.0 / (np1 * np1);
      sum += t[n];
    }

  gsl_sum_levin_u_accel (t, N, w, &sum_accel, &err);

  printf ("term-by-term sum = % .16f using %d terms\n",
          sum, N);

  printf ("term-by-term sum = % .16f using %zu terms\n",
          w->sum_plain, w->terms_used);

  printf ("exact value      = % .16f\n", zeta_2);
  printf ("accelerated sum  = % .16f using %zu terms\n",
          sum_accel, w->terms_used);

  printf ("estimated error  = % .16f\n", err);
  printf ("actual error     = % .16f\n",
          sum_accel - zeta_2);

  gsl_sum_levin_u_free (w);
  return 0;
}
```

The output below shows that the Levin u-transform is able to obtain an estimate of the sum to 1 part in 10^{10} using the first eleven terms of the series. The error estimate returned by the function is also accurate, giving the correct number of significant digits.

```
term-by-term sum =  1.5961632439130233 using 20 terms
term-by-term sum =  1.5759958390005426 using 13 terms
exact value      =  1.6449340668482264
accelerated sum  =  1.6449340669228176 using 13 terms
estimated error  =  0.0000000000888360
actual error     =  0.0000000000745912
```

Note that a direct summation of this series would require 10^{10} terms to achieve the same precision as the accelerated sum does in 13 terms.

33.4 References and Further Reading

The algorithms used by these functions are described in the following papers,

- T. Fessler, W.F. Ford, D.A. Smith, HURRY: An acceleration algorithm for scalar sequences and series *ACM Transactions on Mathematical Software*, 9(3):346–354, 1983. and Algorithm 602 9(3):355–357, 1983.

The theory of the u-transform was presented by Levin,

- D. Levin, Development of Non-Linear Transformations for Improving Convergence of Sequences, *Intern.: J.: Computer Math.* B3:371–388, 1973.

A review paper on the Levin Transform is available online,

- Herbert H. H. Homeier, Scalar Levin-Type Sequence Transformations, http://arxiv.org/abs/math/0005209