CHAPTER

THIRTYONE

NUMERICAL DIFFERENTIATION

The functions described in this chapter compute numerical derivatives by finite differencing. An adaptive algorithm is used to find the best choice of finite difference and to estimate the error in the derivative. These functions are declared in the header file `gsl_deriv.h`.

31.1 Functions

int **gsl_deriv_central** (const *gsl_function* * *f*, double *x*, double *h*, double * *result*, double * *abserr*)

This function computes the numerical derivative of the function `f` at the point `x` using an adaptive central difference algorithm with a step-size of `h`. The derivative is returned in `result` and an estimate of its absolute error is returned in `abserr`.

The initial value of `h` is used to estimate an optimal step-size, based on the scaling of the truncation error and round-off error in the derivative calculation. The derivative is computed using a 5-point rule for equally spaced abscissae at $x - h$, $x - h/2$, x, $x + h/2$, $x + h$, with an error estimate taken from the difference between the 5-point rule and the corresponding 3-point rule $x - h$, x, $x + h$. Note that the value of the function at x does not contribute to the derivative calculation, so only 4-points are actually used.

int **gsl_deriv_forward** (const *gsl_function* * *f*, double *x*, double *h*, double * *result*, double * *abserr*)

This function computes the numerical derivative of the function `f` at the point `x` using an adaptive forward difference algorithm with a step-size of `h`. The function is evaluated only at points greater than `x`, and never at `x` itself. The derivative is returned in `result` and an estimate of its absolute error is returned in `abserr`. This function should be used if $f(x)$ has a discontinuity at `x`, or is undefined for values less than `x`.

The initial value of `h` is used to estimate an optimal step-size, based on the scaling of the truncation error and round-off error in the derivative calculation. The derivative at x is computed using an "open" 4-point rule for equally spaced abscissae at $x + h/4$, $x + h/2$, $x + 3h/4$, $x + h$, with an error estimate taken from the difference between the 4-point rule and the corresponding 2-point rule $x + h/2$, $x + h$.

int **gsl_deriv_backward** (const *gsl_function* * *f*, double *x*, double *h*, double * *result*, double * *abserr*)

This function computes the numerical derivative of the function `f` at the point `x` using an adaptive backward difference algorithm with a step-size of `h`. The function is evaluated only at points less than `x`, and never at `x` itself. The derivative is returned in `result` and an estimate of its absolute error is returned in `abserr`. This function should be used if $f(x)$ has a discontinuity at `x`, or is undefined for values greater than `x`.

This function is equivalent to calling *gsl_deriv_forward()* with a negative step-size.

31.2 Examples

The following code estimates the derivative of the function $f(x) = x^{3/2}$ at $x = 2$ and at $x = 0$. The function $f(x)$ is undefined for $x < 0$ so the derivative at $x = 0$ is computed using *gsl_deriv_forward()*.

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

double f (double x, void * params)
{
  (void)(params); /* avoid unused parameter warning */
  return pow (x, 1.5);
}

int
main (void)
{
  gsl_function F;
  double result, abserr;

  F.function = &f;
  F.params = 0;

  printf ("f(x) = x^(3/2)\n");

  gsl_deriv_central (&F, 2.0, 1e-8, &result, &abserr);
  printf ("x = 2.0\n");
  printf ("f'(x) = %.10f +/- %.10f\n", result, abserr);
  printf ("exact = %.10f\n\n", 1.5 * sqrt(2.0));

  gsl_deriv_forward (&F, 0.0, 1e-8, &result, &abserr);
  printf ("x = 0.0\n");
  printf ("f'(x) = %.10f +/- %.10f\n", result, abserr);
  printf ("exact = %.10f\n", 0.0);

  return 0;
}
```

Here is the output of the program,

```
f(x) = x^(3/2)
x = 2.0
f'(x) = 2.1213203120 +/- 0.0000005006
exact = 2.1213203436

x = 0.0
f'(x) = 0.0000000160 +/- 0.0000000339
exact = 0.0000000000
```

31.3 References and Further Reading

The algorithms used by these functions are described in the following sources:

- Abramowitz and Stegun, *Handbook of Mathematical Functions*, Section 25.3.4, and Table 25.5 (Coefficients for Differentiation).
- S.D. Conte and Carl de Boor, *Elementary Numerical Analysis: An Algorithmic Approach*, McGraw-Hill, 1972.