

# Science and Computers II: Project 2

Project 2 will be assessed based on the material in three parts below. The exercises cover numerical differentiation and numerical integration. Parts I and II are required for project 2. The deadline for project 2 is **Friday April 3, 2008 at 4pm**.

## Part I: Numerical Differentiation

### Exercise 1: Simple numerical derivatives

Implement formulae for finding one-sided and two-sided derivatives of a function  $f(x)$ :

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \approx \frac{f(x+h) - f(x-h)}{2h}.$$

*Empirically* deduce formula for the error term as a function of  $h$ . You should try *a number of different functions* to do this. Illustrate what happens when the step size  $h$  becomes too small or too large. What range of steps,  $h$ , are stable? Does the error fall within the bounds that we discussed in class? What happens to your error when you try to use your code on the functions  $f(x) = x$ ,  $f(x) = x^2$ ,  $f(x) = x^3$ ? Explain this. Finally, compare the results from the above finite difference formulae with the error as a function of  $h$  for the five-point formula

$$f'(x) \approx \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h}.$$

### Exercise 2: More accurate formulae

Implement Richardson Extrapolation for the two-point and three-point formulae above and show how much more accurate the extrapolated result becomes with modest values for the step size  $h$ .

### Exercise 3: Second derivative

Finally, consider how we may evaluate the second derivative of a function numerically. By expanding  $f(x+h)$  in a Taylor series show that the second derivative may be approximated as

$$f''(x) = \frac{f(x+h) + f(x-h) - 2f(x)}{h^2}$$

and deduce an upper bound for the error term as we did in class for the first derivative. Does this error term include just even, just odd or all powers of  $h$ ? Give a proof supporting your answer. Can Richardson Extrapolation be used in this case? Why or why not? Would you expect the determination of numerical second derivatives to be more or less stable than the determination of first derivatives? Explain your answer. Again show how the error depends on  $h$  for a number of simple functions. What range of  $h$  is stable this time?

## Part II: Numerical Integration

This exercise involves the implementation of Romberg's method for one-dimensional quadrature (numerical integration). You should build the up in stages so that you can see the power of the Romberg method demonstrated.

## Exercise 1: Simple implementation of the trapezium and midpoint rules

Write C code to do the following

```
double trap( double (*func)(double x), double a, double b, double h )
```

where a and b are the the end-points of the integration, h is the interval size to be used and func is the name of the function to be integrated, which should have the prototype

```
double my_function( double x )
```

So, for example, to evaluate

$$\int_0^1 \exp(-x^2) dx$$

you would write

```
#include <stdio.h>
#include <math.h>
```

```
double trap( double (*func)(double x), double a, double b, double h )
{
    ...
    /* to evaluate the function at a value x call */
    f = func( x );
    ...
}
```

```
double myfun( double x )
{
    return exp( - (x*x) );
}
```

```
int main( int argc, char *argv[] )
{
    double a = 0.0;
    double b = 1.0;
    double h = 0.1;
    double ans;

    ans = trap( myfun, a, b, h );

    fprintf( stdout, "Answer = %e\n", ans );

    return 0;
}
```

If you are not sure how to pass a function name as an argument to another function, read Kernighan and Ritchie section 5.11 or see me.

Test these routines to ensure that they are working. By using a simple function e.g.  $f(x) = e^x$ , determine the rate of convergence (as a function of  $h$ ) empirically. Also apply your code to the function  $f(x) = \sqrt{x}$ . How is the convergence rate affected? Why is this the case?

### Exercise 2: Some additional complications

**Finite jumps:** Consider the function

$$f(x) = \begin{cases} x^2 - 1 & 0 \leq x \leq 0.5, \\ e^x \sin x & 0.5 < x \leq 1. \end{cases}$$

Evaluate this integral using the composite trapezoidal rule with  $h = 0.05$  and  $h = 0.04$ . Why are the answers so different? Now evaluate the integral in two parts

$$\int_0^1 f(x) dx = \int_0^{0.5} (x^2 - 1) dx + \int_{0.5}^1 e^x \sin x dx$$

What can you learn from this?

**Infinite discontinuities:** A problem occurs for integrals such as

$$\int_a^b \frac{g(x)}{(x-a)^p} dx$$

where  $g(x)$  is well behaved at  $x = a$  and  $p < 1$ . What happens to the error formula for composite trapezoidal integration for a function like this? What is a bound on the error? This problem can be alleviated by expanding  $g(x)$  as a Taylor series around  $x = a$

$$P(x) = \sum_{k=0}^n \frac{g^{(k)}(a)}{k!} (x-a)^k.$$

The integral of  $f$  can now be approximated as

$$\int_a^b \frac{g(x) - P(x)}{(x-a)^p} dx + \int_a^b \frac{P(x)}{(x-a)^p} dx$$

where the first term now has  $n$  bounded derivatives at  $x = a$  and the second term can be computed analytically. For an efficient implementation of the trapezium or midpoint rule, what should  $n$  be chosen to be?

Approximate the integral

$$\int_0^1 \frac{\exp(x)}{\sqrt{x}} dx$$

using this method and the composite trapezium method. What do you learn?

**Infinite limits:** Consider the integral

$$\int_0^\infty f(x) dx.$$

Use the change of variable  $t = 1/x$  to remove the infinite limit. Use the composite midpoint rule together with this change of variable to evaluate

$$\int_1^{\infty} x^{-2} \sin x \, dx$$

What do you think could be done with an integral with two infinite limits?

### Exercise 3: Better implementation of the trapezium rule

So far, we have used integration formulas without any demand that a certain accuracy be achieved. The simplest method for guaranteeing a given accuracy with the trapezoidal rule is just to start off using two points, treating the integration range as a single interval. The integral should then be subdivided into two intervals and the trapezoidal rule used again, then subdivided into four, and so on, continuing until the result has converged. This is quite efficient as at any stage, the evaluations of the function at the previous points can be reused.

Write C code to implement the trapezoidal rule in this way being careful to avoid making unnecessary function evaluations. In particular, you should write code for a function which expects the following arguments:

```
double trap( double (*func)(double), double a, double b,
            double tol, int *neval )
```

On entry `a` and `b` should contain the endpoints of the integral and `func` should contain the name of a function being integrated. `tol` should be set to the absolute accuracy to which the integral is required to be evaluated. The function should return the result and set `neval` to return the number of times `func` was called.

### Exercise 4: Romberg Integration

The principle of Romberg integration was introduced in class. It was shown that the error made by the trapezium rule was a power series that is entirely even in the step size  $h$ . In other words, we may write

$$\int f(x) \, dx = I(h) + k_1 h^2 + \mathcal{O}(h^4)$$

and similarly

$$\int f(x) \, dx = I(h/2) + k_1 (h/2)^2 + \mathcal{O}(h^4)$$

from which it may be deduced that

$$\int f(x) \, dx = \frac{4I(h/2) - I(h)}{3} + \mathcal{O}(h^4)$$

so that the leading order error term is canceled. In Romberg integration, this procedure is continued with the step size being halved again and a suitable combination chosen to cancel off the next term  $\mathcal{O}(h^4)$  and so on.

Write C code to implement this procedure. A suitable form for it may be

```
double romberg( double (*func)(double), double a, double b,
              double tol, int *neval )
```

where the variables have the same meaning as the previous section.

Evaluate the integral

$$\int_0^\pi \sin x \, dx$$

using your trapezium and Romberg programs. Which one takes requires the fewest evaluations of the integrand and by what factor? Choose an absolute tolerance of  $10^{-10}$ .

### Exercise 5: The advantages of thinking

Sometimes a simple manipulation of the integrand can drastically reduce the work the computer has to do. Consider, for example, the evaluation of the integral

$$\int_\alpha^1 \frac{dx}{e^x - 1}$$

where  $\alpha = 10^{-6}$ . Evaluate this using Romberg integration to an accuracy of say  $10^{-12}$ . Why are so many evaluations of the function required?

Consider evaluating the integral as follows:

$$\int_\alpha^1 \frac{dx}{e^x - 1} = \int_\alpha^1 \left[ \frac{1}{e^x - 1} - \frac{1}{x} \right] dx + \int_\alpha^1 \frac{dx}{x}$$

and using your program to evaluate the first integral and doing the second one analytically. How many evaluations of the function were required this time? Explain the difference. Clearly, for small values of  $x$  a delicate cancellation occurs in the first term. Will this procedure of splitting the integral into two parts be stable as  $\alpha$  becomes smaller? What may go wrong in this specific case?