AMTH142
Lecture 18

Numerical Integration I

Contents

1 Some Basic Methods 2
  1.1 The Midpoint Rule 2
  1.2 The Trapezoidal Rule 5
  1.3 Simpson’s Rule 7

2 Rounding Errors 9
1 Some Basic Methods

In this lecture we will look at the numerical evaluation of integrals

$$I(f) = \int_a^b f(x)dx$$

Almost every method of numerical integration uses an approximation of the form

$$I(f) \approx \sum_{i=1}^{n} w_i f(x_i)$$

where the points $x_i \in [a,b]$ at which the function is evaluated are called nodes and the multipliers $w_i$ are called weights.

1.1 The Midpoint Rule

Divide the interval $[a,b]$ into $n$ equal subintervals of length $h = (b-a)/n$. Let the nodes $x_i, i = 1, \ldots, n$ be the midpoint of the $i$-th interval and approximate the integral by the areas of the rectangles with base $h$ centered on $x_i$ and height $f(x_i)$.

This gives the approximation

$$\int_a^b f(x)dx \approx h \sum_{i=1}^{n} f(x_i)$$

which is easily implemented in Scilab:
function y = midint(f, a, b, n)
    h = (b-a)/n
    x = linspace(a+h/2, b-h/2, n)
    y = h*sum(f(x))
endfunction

As our example we will evaluate the integral
\[ I = \frac{2}{\sqrt{\pi}} \int_0^1 e^{-x^2} \, dx \]
associated with the normal distribution. The function
\[ \text{Erf} \,(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} \, dt \]
is called the error function and is known to Scilab.

An accurate value of our integral is
-->ii = erf(1)
ii =

0.8427008

We will evaluate the integral using the midpoint rule with \( n = 2, 4, 8, \ldots, 1024 \) points and examine the error:

-->function y = func(x)
--> y = 2*exp(-x.*x)/sqrt(%pi)
-->endfunction

-->intf = zeros(1,10);

-->for k = 1:10
-->    intf(k) = midint(func, 0, 1, 2^k);
-->end

-->intf
intf =

column 1 to 5

3
We see that when $n$ increases by a factor of 2, so that $h$ decreases by a factor of 2, the error decreases by a factor of 4, which suggests that the error is proportional to $h^2$. This is supported by the log-log plot of the error.
1.2 The Trapezoidal Rule

Again divide the interval \([a, b]\) into \(n\) equal subintervals of length \(h = (b - a)/n\). Let the nodes \(x_i, i = 0, \ldots, n\) be endpoints of the intervals and approximate the integral by the linearly interpolating the function values at the nodes and taking the areas of the resulting trapeziums.

The area of the trapezium with base \([x_i, x_{i+1}]\) is

\[
\frac{[f(x_i) + f(x_{i+1})]}{2} h
\]

giving the approximation

\[
\int_a^b f(x) dx \approx \frac{h}{2} \left( [f(x_0) + f(x_1)] + [f(x_1) + f(x_2)] + [f(x_2) + f(x_3)] + \cdots + [f(x_{n-1}) + f(x_n)] \right)
\]

\[
= \frac{h}{2} (f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(x_n))
\]

Again, this is easy to implement in Scilab:

```scilab
function y = trapint(f, a, b, n)
    h = (b-a)/n
    x = linspace(a+h, b-h, n-1)
    y = h*(f(a)/2 + sum(f(x)) + f(b)/2)
endfunction
```

5
We will perform the same calculations as for the midpoint rule:

```latex
\texttt{--for } k = 1:10 \\
\texttt{-- inte(k) = trapint(func, 0, 1, 2^k); \\
\texttt{--end}

\texttt{--inte}
\texttt{inte} =

\begin{verbatim}
  column 1 to 5
  ! 0.8252630  0.8383678  0.8416192  0.8424305  0.8426332 !
  column 6 to 10
  ! 0.8426839  0.8426966  0.8426997  0.8427005  0.8427007 !
\end{verbatim}

```

```latex
\texttt{--ef = abs(inte-ii)}
\texttt{ef} =

\begin{verbatim}
  column 1 to 5
  ! 0.0174378  0.0043330  0.0010816  0.0002703  0.0000676 !
  column 6 to 10
  ! 0.0000169  0.0000042  0.0000011  2.639E-07  6.598E-08 !
\end{verbatim}

```

```latex
\texttt{--plot2d(log10(ef))}
```

The results are very similar to those for the midpoint rule. The error is again proportional to $h^2$ but, for this function, the error for a given value of $h$ is about twice error in the midpoint approximation.
1.3 Simpson’s Rule

Once more we divide the interval \([a, b]\) into \(n\) equal subintervals of length \(h = (b-a)/n\). Again the nodes \(x_i, i = 0, \ldots, n\) are endpoints of the intervals. For Simpson’s rule we assume that \(n\) is even and we interpolate the function values over \(pairs\) of subintervals by a quadratic polynomial.

![Diagram](image)

The integral is approximated by the area under the interpolating quadratics. A simple calculation shows that this area is

\[
\frac{h}{6} [f(x_i) + 4f(x_{i+1}) + f(x_{i+2})]
\]
giving the approximation

\[
\int_a^b f(x)dx \approx \frac{h}{6} \left( \left[ f(x_0) + 4f(x_1) + f(x_2) \right] + \left[ f(x_2) + 4f(x_3) + f(x_4) \right] + \cdots + \left[ f(x_{n-2}) + 4f(x_{n-1}) + f(x_n) \right] \right)
\]

By considering the odd and even nodes separately, we see that Simpson’s approximation with \(n\) subintervals, \((n\ even),\) is the sum of \(2/3\) times the midpoint approximation with \(n/2\) subintervals plus \(1/3\) times the trapezoidal approximation with \(n/2\) subintervals. We can use this to give a simple Scilab implementation of Simpson’s method:

```
function y = simpint(f, a, b, n)
    y = (2*midint(f, a, b, n/2) + trapint(f, a, b, n/2))/3
endfunction
```

Using the same test as before:

```plaintext
-->for k = 1:10
-->  intf(k) = simpint(func, 0, 1, 2^k);
-->end

-->intf
intf  =

column 1 to 5

! 1.2192292  0.8427361  0.8427030  0.8427009  0.8427008 !

column 6 to 10

!  0.8427008  0.8427008  0.8427008  0.8427008  0.8427008 !

-->ef = abs(intf-ii)
ef  =
column 1 to 5
! 0.376528 0.0000353 0.0000022 1.406E-07 8.795E-09 !

column 6 to 10

-->plot2d(log10(ef))

The first approximation, \( n = 2 \), is actually incorrect since the implementation of the trapezoidal rule doesn’t work for \( n = 1 \) (the problem is with \texttt{linspace}, why?) The error in Simpson’s method decreases by a factor of 16 when \( h \) decreases by a factor of 2, indicating that the error is proportional to \( h^4 \), a substantial improvement of the midpoint and trapezoidal methods.

2 Rounding Errors

Numerical integration does not present any problems as far as rounding errors are concerned. In general, we expect a relative error due to rounding to be
the order of $\varepsilon_{\text{mach}}$ for any numerical integral. To see why we will return to the general form of the numerical approximation

$$\int_a^b f(x) \, dx \approx \sum_{i=1}^n w_i f(x_i)$$

Rounding errors will arise from the additions, multiplications by the $w_i$ and in the function evaluations. In fact we can think of the general formula as repeatedly performing the operations (1) evaluate $f(x_i)$, (2) multiply by $w_i$, and (3) add. The sequence of operations will produce a rounding error of order

$$e_i \approx w_i f(x_i) \varepsilon_{\text{mach}}$$

and the total rounding error will be of order

$$e \approx \sum_{i=1}^n w_i f(x_i) \varepsilon_{\text{mach}}$$

$$\approx \varepsilon_{\text{mach}} \sum_{i=1}^n w_i f(x_i)$$

$$\approx \varepsilon_{\text{mach}} \int_a^b f(x) \, dx$$

giving a relative error in the integral of $\varepsilon_{\text{mach}}$.

As an demonstration, we will show that Simpson’s rule can evaluate our example integral to an accuracy of order $\varepsilon_{\text{mach}}$. From the previous section we see that for $n = 1024$ we have an error of about $10^{-14}$. Doubling $n$ will decrease the error by a factor of 16 which will bring it down to the order of machine epsilon:

---> ef = simpint(func, 0, 1, 2048)
    ef =

    0.8427008

---> ef - ii
    ans =

    0.