# MACSI One Day Graduate Course: Numerical Solution to Differential Equations using Matlab

## **Part 4: Verification of the rates of convergence**

Niall Madden and Meghan Stephens
Department of Mathematics

National University of Ireland, Galway
Ollscoil na hÉireann, Gaillimh

3rd of April 2007.

We want to verify that our program and numerical method yield the expect result:

There is a constant $C$ that does not depend on $N$ such that

$$\|u - U\| \leq CN^{-2}.$$

# A simple problem

Consider the problem:

$$-u''(x) + u(x) = 1 + x \text{ on } (0, 1), \qquad u(0) = u(1) = 0.$$

The solution to this is

$$u(x) = 1 + x - \left(e^{-x}(e^2 - 2e) + e^x(2e - 1)\right)/(e^2 - 1).$$

We'll use this to test the code.

# A simple problem

Download and run the program **TestError.m**

Two new Matlab functions:

- **fprintf** works very similarly to **printf** in C.

```
fprintf(' %5d | %10.3e | %7.2e \n', N, Error, C);
```

  - Display the integer **N**, padding up to 5 spaces,
  - Display the double **Error** in *exponential notation* (using a lowercase e as in 3.1415e+00) filling up to 10 spaces, and with 3 digits to the right of the decimal point.
  - **\n** prints a new line.

- **subplot(A, B, C)** plot the next figure in the $C$th position of an array of $A \times B$ figures.

# Computing rates of convergence

To verify that we get the correct rate of convergence, use the following calculation:

- denote by $U^N$ the solution computed on a mesh with $N$ intervals.
- Let $\mathcal{E}_N = \|u - U^N\|$
- Suppose that $\mathcal{E} \sim CN^{-\gamma}$. Then

$$\frac{E_N}{E_{2N}} \approx 2^\gamma, \text{ and so } \gamma \approx \log_2(E_N/E_{2N})$$

Download and try **`TestRates.m`**

As you'll notice from the code, we can't compute the rate of convergence for the first value of **N**.

So our script behaves differently in certain cases. This is achieved using an **if** block:

```
if (k>2)
   Rate(k) = log2( Error(k-1)/Error(k));
   fprintf(' %5d | %9.3e | %7.2e | %5.2f \n', ...
       N, Error(k), C(k), Rate(k));
else
   fprintf(' %5d | %9.3e | %7.2e | \n', ...
       N, Error(k), C(k));
end
```

See **doc if** for more information.

# For harder problems

Generally, we don't have the exact solution to the problem we want to solve.

If we did, we wouldn't need a numerical method!

In this case, take one of two approaches:

- Compute the solution on the finest mesh (i.e., largest $N$) that your computer can handle. Verify that the solutions for smaller $N$ converge towards this one.
- Compare a computed solution on $N$ intervals, with the solution computed on $2N$ intervals.
  With a little work, one can show (mathematically) that the estimates for the rate of convergence are reasonable.

# For harder problems

The first of these approaches is taken in `TestRates2.m`

Note the use of the `interp1` function. This evaluates the piecewise linear interpolant to the best approximation of $u$ on the coarser mesh.

This is useful, particularly for nonuniform meshes.

Other interpolants could be used, e.g, cubic splines, polynomials, etc. But use these with caution: they may given spurious results.

# Why we need to verify the code

We'll now develop code for the more general problem

$$-\varepsilon u''(x) + qu'(x) + ru(x) = f(x).$$

To *discretize* $u' = \frac{du}{dx}$ we'll use the 2nd order central difference operator

$$D^c u_i := \frac{1}{2h}\Big(u_{i+1} - u_{i-1}\Big).$$

Download the programs `Central_Diff_BVP.m` and `Test_Central_Diff.m`

Note that we use the variable name **`epsilon`**. **Never** use the variable name **`eps`** – it is used to store the *machine epsilon*: the distance from 1.0 to the next largest double-precision number.

# Why we need to verify the code

Download the files `Central_Diff_BVP.m` and `Test_Central_Diff.m`

Run the test. All should appear well; we get the expected rate of convergence.

But observe what happens when we take $\varepsilon$ smaller, e.g., **`epsilon=1e-2`** Note that there are oscillations present in the computed solution and that, for small $N$ the rates of convergence are less than one would expect.

Now reduce $\epsilon$ further: the problem becomes more dramatic.

# Why we need to verify the code

There are various ways of explaining this phenomenon. One way is by observing that the operator:

$$Lu(x) := -\varepsilon u''(x) + q(x)u'(x) + r(x)u(x)$$

obeys as maximum principle, the discrete one

$$L^h U_i := -\varepsilon \delta^2 U_i + q(x_i) D^c U_i + r(x_i) U_i.$$

does not, unless $h \leq 2\varepsilon$, which is very restrictive.

# Why we need to verify the code

One way to circumvent this problem is to use a different discretization of $u'(x)$:

$$D^{-}U_i := \frac{1}{h}\big(U_i - U_{i-1}\big).$$

Now the associated difference operator will satisfy a max prin.

But the price we pay for accuracy is in the rate of convergence; with this method

$$\|u - U\| \le CN^{-1}$$

**Exercise**

Implement this method, and verify that, for small $\varepsilon$ it is only 1st-order accurate.