Consider the reaction-diffusion equation:

$$-\frac{d^2u}{dx^2} + r(x)u = f(x), \quad 0 < x < 1 \quad \text{with} \quad u(0) = 0,\ u(1) = 0.$$

with $r = 100$ and $f = 1$. Writing $r = \omega^2$, so that $\omega = 10$, the exact solution (given in lectures) is:

$$\frac{1}{100} - \frac{(\exp(10x) + \exp(10(1-x)))}{100(1+\exp(10))}.$$

Suppose we choose $N = 4$ intervals on $[0, 1]$ and set $x_0 = 0$, $x_1 = \frac{1}{4}$, $x_2 = \frac{1}{2}$, $x_3 = \frac{3}{4}$ and $x_4 = 1$. Let U_j denote an approximation to the exact solution $u(x)$ at $x = x_j$, and denote $r_j = r(x_j)$ and $f_j = f(x_j)$. Now apply the centered finite difference scheme:

$$-\frac{1}{h^2}U_{j-1} + \left(\frac{2}{h^2} + r_j\right)U_j - \frac{1}{h^2}U_{j+1} = f_j, \quad j = 1:3.$$

Since $h = \frac{1}{4}$ and the boundary conditions give $U_0 = 0$ and $U_4 = 0$, we obtain three equations for the unknown values U_1, U_2, U_3:

$$\begin{array}{rcl} 132U_1 - 16U_2 & = & 1 \\ -16U_1 + 132U_2 - 16U_3 & = & 1 \\ -16U_2 + 132U_3 & = & 1 \end{array} \Rightarrow \begin{pmatrix} 132 & -16 & 0 \\ -16 & 132 & -16 \\ 0 & -16 & 132 \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \\ U_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

Solving this 3×3 linear system (using Gaussian Elimination by hand or in MATLAB using trisolve.m) gives: $U_1 = 0.008751$, $U_2 = 0.009697$, $U_3 = 0.008751$.

In the figure below, we plot the exact and approximate solution obtained using $N = 4$ intervals and then increase N to $N = 8$, $N = 16$ and $N = 32$ intervals.

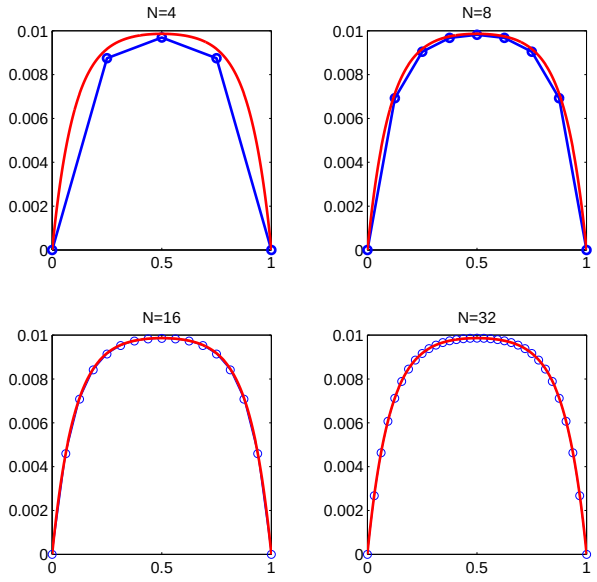

Figure 1: Exact solution (solid line) and numerical approximations (solid line with circles) to the solution with $N = 4, 8, 16, 32$ intervals

To reproduce the above graphs, download the MATLAB m-files `trisolve.m` and `reac_diff_1D.m` from the course webpage. Save these files to your P: drive. Start up MATLAB and change directory so that your current working directory is your P: drive. Open the file `reac_diff_1D.m` in the editor window - or else just type

```
>> type reac_diff_1D
```

at the MATLAB prompt. Read the file to understand what it does. To run the program, you type:

```
[u,u_exactx,error]=reac_diff_1D(w,N)
```

at the prompt in the command window, inserting the desired values for w, (where $r = w^2$) and N, the number of intervals for the finite difference calculation. For the above example $w = 10$. Hence, typing,

```
[u,u_exactx,error]=reac_diff_1D(10,4)
[u,u_exactx,error]=reac_diff_1D(10,8)
[u,u_exactx,error]=reac_diff_1D(10,16)
[u,u_exactx,error]=reac_diff_1D(10,32)
```

will produce the results above. Try it!

As we increase the number of intervals, the approximate solution is able to resolve the steep layers in the solution close to $x = 0$ and $x = 1$. Let's consider what happens to the error for a fixed value $x = 0.5$ (say) as we increase the number of intervals. The exact solution at $x = 0.5$ is $u = 0.00986524$ (to 8 dp). In the table below, we list the approximations obtained to this value for $N = 4, 8, 16$ and 32 intervals. In the case $N = 4$, the value we are interested in is U_3. For the case $N = 8$, we want U_5 etc..

N	$u(0.5)$	j	U_j	$\lvert u(0.5) - U_j \rvert$	$\max_{j=1:N} \lvert e_j \rvert$
4	0.00986524	3	0.00969726	1.67991e-004	4.22474e-004
8	0.00986524	5	0.00982192	4.33318e-005	2.07740e-004
16	0.00986524	9	0.00985432	1.09302e-005	5.66227e-005
32	0.00986524	17	0.00986251	2.73908e-006	1.48398e-005

The absolute value of the error $e_j = u(0.5) - U_j$ is listed in the 5th column. Notice that the error at the point $x = 0.5$ is **decreasing by roughly a factor of four each time we double the number of intervals or equivalently reduce h by a factor of two**. If we keep on increasing the number of intervals, we can approximate the value $u(0.5)$ to whatever degree of accuracy is required. In the 6th column, the maximum value of the error over all points used for a fixed number of intervals is listed. Notice that this quantity also decreases by roughly a factor of four each time we double the number of intervals. Clearly, the method is convergent. **You should relate these observations to the Theorems stated in the lecture notes for this method.**

Note: you will not be expected to solved a big tri-diagonal system of equations by hand in an exam. However, you can do it for examples with small N and you should be able to derive the tridiagonal system of equations for any choice of $r(x)$, $f(x)$, N, and for different boundary conditions.

Download the MATLAB code `trisolve.m` and `reac_diff_1D.m` from the course webpage and try and reproduce the results above for yourself!