Scientific Computing for DPhil Students II
Assignment 3 Solutions

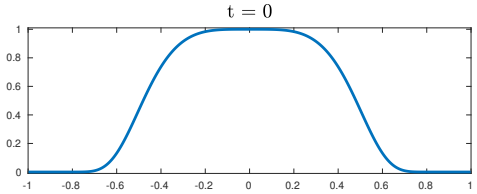
1. Here is my code, adapted from `m44_CrankNicolson.m`. Plots are shown for $t = 0$ and $t = 0.125$.

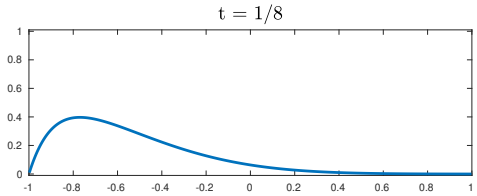
```
% a3_1.m - modification of m44_CrankNicolson.m to solve advection-diffusion
%          problem u_t = u_xx + 10u_x, u(-1)=u(1)=0

% Solve PDE for various step sizes:
  uuvec = [];
  for logh = 5:9
    h = 2^(-logh);
    x = (-1+h:h:1-h)'; k = .25*h;
    u = exp(-10*x.^4./(1-x.^2));
    L = length(x);
    % sparse matrix for implicit terms:
      a = (1+k/h^2); b = -.5*k/h^2+10*.25*k/h; c = -.5*k/h^2-10*.25*k/h;
      main = a*sparse(ones(L,1));
      subdiag = b*sparse(ones(L-1,1));
      superdiag = c*sparse(ones(L-1,1));
      B = diag(main) + diag(superdiag,1) + diag(subdiag,-1);
    % sparse matrix for explicit terms:
      a = (1-k/h^2); b = .5*k/h^2-10*.25*k/h; c = .5*k/h^2+10*.25*k/h;
      main = a*sparse(ones(L,1));
      subdiag = b*sparse(ones(L-1,1));
      superdiag = c*sparse(ones(L-1,1));
      A = diag(main) + diag(superdiag,1) + diag(subdiag,-1);
    tmax = 1/8; nsteps = tmax/k;
    hold off, shg
    plt = plot(x,u,'linewidth',2); title('t = 0','fontsize',14)
    axis([-1 1 -.01 1.01]), grid, pause
    for step = 1:nsteps
      u = B\(A*u);                        % Crank-Nicolson
      set(plt,'ydata',u), drawnow
    end
    uu = u(2^logh/4), uuvec = [uuvec; uu];
    title('t = 1/8','fontsize',14), pause
  end

% Richardson extrapolation to find value U(x=-.75,t=0.125):
%    (the correct value seems to be about 0.395655846)
  a = uuvec;
  b = a(2:end) + (a(2:end)-a(1:end-1))/3; b = [NaN; b];
  c = b(2:end) + (b(2:end)-b(1:end-1))/15; c = [NaN; c];
  disp([a b c])
```

The assignment asked for $u(x = -0.75, t = 0.125)$ to at least four digits of accuracy. This is easily found to be 0.3957. One can get many more digits of accuracy with Richardson extrpolation (not required in this assignment), as carried out by the final lines of the code. The result printed is

```
   0.3988985511            NaN            NaN
   0.3964554067   0.3956410252            NaN
   0.3958550496   0.3956549305   0.3956558576
   0.3957056041   0.3956557890   0.3956558462
   0.3956682828   0.3956558423   0.3956558458
```

It would seem that we have $u(x = -0.75, t = 0.1250) \approx 0.395655846.$





This problem can also be solved in Chebfun with PDE15S or CHEBGUI.

2. (a) To analyze stability of Richardson's formula

$$\frac{v_j^{n+1} - v_j^{n-1}}{2k} = \frac{v_{j+1}^n - 2v_j^n + v_{j-1}^n}{h^2}$$

we consider a solution of the form $v_j^n = g(\xi)^n e^{i\xi jh}$, where ξ is an arbitrary real wave number and $g(\xi)$ an associated *amplification factor*. Inserting this ansatz in the finite difference formula gives

$$g(\xi) - g(\xi)^{-1} \;=\; \frac{2k}{h^2}(e^{i\xi h} - 2 + e^{-i\xi h}) \;=\; \frac{2k}{h^2}(e^{i\xi h/2} - e^{-i\xi h/2})^2 \;=\; -\frac{8k}{h^2}\sin^2(\xi h/2).$$

In other words, $g(\xi)$ satisfies the quadratic equation

$$g(\xi)^2 + 2bg(\xi) - 1 = 0, \qquad b = \frac{4k}{h^2}\sin^2(\xi h/2),$$

with solution

$$g(\xi) \;=\; -b \pm \sqrt{b^2+1}.$$

This attains its largest absolute value for $\xi = \pi/h$, corresponding to a sawtoothed mode on the grid:

$$g = \max_\xi |g(\xi)| = b + \sqrt{b^2+1}, \qquad b = \frac{4k}{h^2}.$$

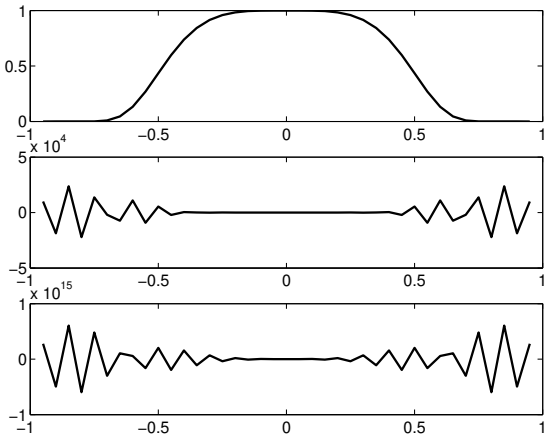
Now it is clear that no matter how small k is, g is bigger than 1. Even in the limit $k/h^2 \to 0$ we find $g \sim 1 + 4k/h^2$, which may be close to 1 but will compound after t/k steps to $\sim \exp(4t/h^2)$—which is, to put it mildly, large. Thus Richardson's formula is explosively unstable. For the particular parameters $h = 0.05$, $k = 0.001$ of part (b), we have $b = 1.6$ and

$$g \;\approx\; 3.49.$$

(b) Here is my code. It prints $G = 3.31$, which agrees pretty well with g.

```
% a3_2.m - Assmt. 3, problem 2: leap frog for heat equation.
%    This code combines elements of m41_implicit.m and m45_leapfrog.

% Initialize and plot step 0:
  h = 0.05; k = 0.001; x = (-1+h:h:1-h)';
  uold = exp(-10*x.^4./(1-x.^2));
  D = h^(-2)*toeplitz([-2 1 zeros(1,length(x)-2)]);
  subplot(3,1,1), plot(x,uold)

% Compute step 1 by Crank-Nicolson:
  I = eye(length(x));
  A = I + k*D/2; B = I - k*D/2;
  u = B\(A*uold);

% Compute and plot steps 20 and 40:
  A = 2*k*D;
  for i = 2:20, unew = uold + A*u; uold = u; u = unew; end
  umax20 = max(abs(u));
  subplot(3,1,2), plot(x,u)
  for i = 21:40, unew = uold + A*u; uold = u; u = unew; end
  subplot(3,1,3), plot(x,u)
  umax40 = max(abs(u)); G = (umax40/umax20)^(1/20)
```

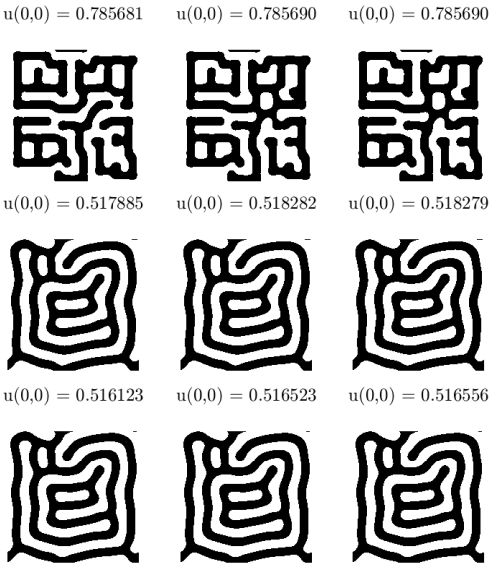
Here is the plot it produces. Note the vertical scales.



3. *The Gray-Scott equations.* Here is the code:

```
function u00 = gs(N);
tic, S = spinop2('gs');
for i = 1:3
  dt = 2^(3-i);
  u = spin2(S,N,dt,'plot','off');
  subplot(1,3,i)
  plot(u{1}-.5,'zebra'), axis square off
  u00 = u{1}(0,0);
  s = sprintf('u(0,0) = %8.6f\n',u00);
  title(s,'fontsize',12), drawnow
end
toc
```

(a) Running `gs(32)`, `gs(64)`, and `gs(128)` gives these results. On my desktop these runs take 5, 14, and 49 seconds.



(b) The code is running Chebfun's `spin2` code to solve the Gray-Scott equations by the ETDRK4 exponential integrator method. We can see some of the details like this:

```
S = spinop2('gs')

S = spinop2 with properties:

     domain: [0 1 0 1]
       init: [2InfInf chebfun2v]
        lin: @(u,v)[2e-5*lap(u);1e-5*lap(v)]
     nonlin: @(u,v)[F*(1-u)-u.*v.^2;-(F+K)*v+u.*v.^2]
      tspan: [0 5000]
    numVars: 2
```

So the equation is

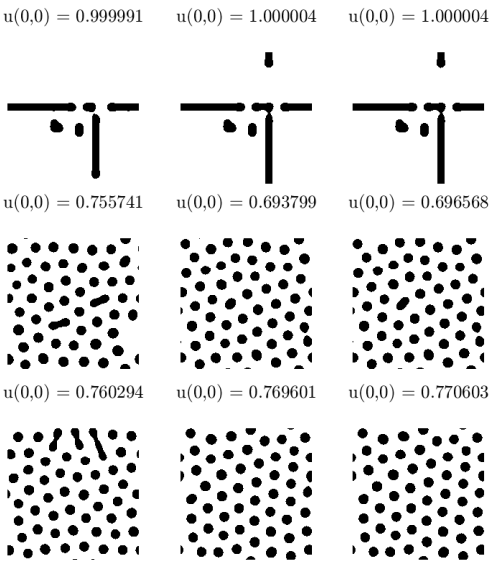
$$u_t = 0.00002\Delta u + F(1-u) - uv^2, \quad v_t = 0.00001\Delta v - (F+K)v + uv^2.$$

This display, however, doesn't tell us the values of the parameters F and K. To find these numbers, we can look in the spinop2 code: they are $F = 0.030$ and $K = 0.057$. The "zebra" output shows the u component, white where $u > 0.5$ and black where $u < 0.5$.

(c) In this array of plots, $k = \Delta t$ is halved as you move right, and the ETDRK4 algorithm should have temporal errors $O(k^4)$. This is consistent with the numbers in the third row. It will be interesting to see if some students explore more carefully.

The spatial errors we expect to be exponential: $O(C^{-N})$ for some $C > 1$. Here again I will be interested to see what people find. In any case to three digits the answer looks like 0.517 and we get this with $N = 128$ and $k = 2$. A somewhat smaller value like $N = 100$ is also good enough.

(d) Now the values are $F = 0.027$ and $K = 0.059$.



(e) Here is a code and the output, a pretty mix of spots and rolls.

```
function u00 = gs(N);
S = spinop2('gs');
F1 = .030; F2 = .027; K1 = .057; K2 = .059;
F = .25*F1 + .75*F2; K = .25*K1 + .75*K2;
S.nonlin = @(u,v) [F*(1-u)-u.*v.^2; -(F+K)*v+u.*v.^2];
for i = 1:3
  dt = 2^(3-i);
  u = spin2(S,N,dt,'plot','off');
  subplot(1,3,i)
  plot(u{1}-.5,'zebra'), axis square off
  u00 = u{1}(0,0);
  s = sprintf('u(0,0) = %8.6f\n',u00);
  title(s,'fontsize',12), drawnow
end
```