Show and tell with PETSc (*revised/actual*)

What is PETSc? It is the *Portable, Extensible Toolkit for Scientific computing*, an open and free C library of numerical software:

http://www.mcs.anl.gov/petsc/

PETSc co-evolved with MPI (= *Message Passing Interface*), also from the Dept. of Energy's Argonne National Laboratory, starting in about 1990, as the fundamental infrastructure for doing science and engineering simulations/computations on the then-new generation of multiple-instruction-multiple-data (MIMD) supercomputers. That is, MPI and PETSc are core "software stack" for parallel computation on supercomputers, the largest of which have (circa 2017) about 10^6 processors (cores).

I am in the midst of writing a book, at the graduate level for computational mathematics, called *PETSc for Partial Differential Equations*. With luck it will be published by SIAM Press in 2018. The C codes for the book's examples are here:

https://github.com/bueler/p4pdes

Two examples from my book.

1. A pair of *coupled diffusion-reaction equations*, which generate patterns, on $(x, y) \in (0, 2.5) \times (0, 2.5)$ and $t > 0$:

$$u_t = D_u \nabla^2 u - uv^2 + \phi(1 - u)$$
$$v_t = D_v \nabla^2 v + uv^2 - (\phi + \kappa)v$$

where D_u, D_v, ϕ, κ are constants. An example run of the C/PETSc code:

```
$ cd c/ch5/ && make pattern
$ mpiexec -n 4 ./pattern -da_refine 5 -ts_monitor -snes_monitor \
    -ts_dt 10 -ts_final_time 5000 -ts_adapt_type none \
    -ts_type beuler -ts_monitor_solution draw
```

The initial condition is four dots in the middle. The spatial derivatives are approximated with a 9-point-stencil version of the usual centered finite difference scheme. The time-stepping is Backward Euler, but it could be the trapezoid rule or an adaptive explicit scheme, etc., as chosen at run-time.

2. The *advection equation* on $(x, y) \in (-1, 1) \times (-1, 1)$ and $t > 0$:

$$u_t + \nabla \cdot (\mathbf{a}u) = 0$$

where $\mathbf{a}(x, y) = (2y, -2x)$. An example run of the C/PETSc code:

```
$ cd c/ch9/ && make advect
$ mpiexec -n 4 ./advect -da_refine 4 -ts_monitor_solution draw -ts_monitor \
   -ts_rk_type 2a -adv_problem rotation -ts_final_time 3.1415926
```

The initial condition $u(x, y, 0)$ would look like a cone and a square tower if you did a surface plot. The spatial derivatives are approximated with finite differences and a "flux-limiting upwind scheme." The time-stepping is by RK2, which is quite suitable for such hyperbolic problems. The time-dependent solution rotates the initial picture. We see that numerical diffusion causes the sharp edges to smooth out.