Topics in simulation of vortex shedding

The vortex-shedding examples 16 and 17 (`example16.py` and `example17.py`) include a study of several aspects of vortex-shedding. Topics include:

- Use of $P^2 - P^1$ Lagrange elements
- Relatively coarse mesh
- Linearized skew-symmetric convection
- Solution methods

Included below are several approaches to studying these issues by investingating alternative or expanded problems. Choose any *two* of these for your project.

1 Other elements

Both of the examples use the $P^2 - P^1$ element pair. This pair is known to be stable in the Ladyzhenskaya-Babuška-Brezzi sense. Other stable elements are presented in Chapter 20, where a careful study of the accuracy and efficiency of several types of elements for solving the Stokes problem is presented.

Consider at least the two elements: Crouzeix-Raviart and lowest-order MINI elements. Feel free to consider other possibilities as well. Apply them to the vortex-shedding problem in Example 16, with the mesh used there. You may have to modify the values of Δt or ν in order to get a good solution. Consider the following questions:

1. How many velocity and pressure degrees of freedom does each of the three methods require?
2. Do they all work with the same values of Δt and ν?
3. Rank the methods in terms of speed. You can use the Python `time` module to measure the total time for a problem. Do not include "JIT" compilation times when you measure running times.
4. Visually examine the solutions. Do they "look the same"?
5. Generate plots of vertical velocity behind the cylinder *versus* time, as was done in the example.
   - Do the three elements result in similar plots?
   - Do the elements result in the same amplitude of oscillation after reaching a steady value? If not, rank them.
   - Do the elements reach a steady oscillation amplitude in about the same amount of problem time? If not, rank them.

Do at least one uniform mesh refinement (`mesh=refine(mesh)`), choosing the finest mesh you feel you can "afford." Answer the above questions again with the finer mesh.

2 Other solvers

Starting with Example 16 as embodied in `example16.py`, make the problem take more time by choosing a final time so that you perform precisely 100 time steps. Perform enough uniform mesh refinements (`mesh=refine(mesh)` so that the overall problem takes several minutes on your computer.

2.1 Krylov solver

The direct (lu) solver used in `example16.py` arrives at machine accuracy (depending on condition number). Regard this solution as correct, and use the difference between this solution (at the end of 100 time steps) as a measure of accuracy. (Error= $\|u - u_{\mathrm{lu}}\|/\|u_{\mathrm{lu}}\|$.)

Solve the same problem using GMRES. What values of convergence criteria (relative and absolute accuracy) are needed to achieve an error smaller than 10^{-9}? How long does it take? You can use the Python `time` module to measure total time, but do not include "JIT" compilation times in your total.

Can you speed up your running time by choosing the previous time step's solution as initial guess for this time step's iteration instead of using a zero intitial guess?

You have a selection of precondiioners available. Try at least three preconditioners in addition to the default "ilu" preconditioner. Rank them on the basis of total time.

3 Stability

There are two different methods presented for convection: traditional and skew-symmetric. You will study the effect of choosing one or the other on stability of the numerical problem.

The solution in `example15.py` reaches a "steady" oscillatory solution by about $t = 3$ in the problem. Call this the "time to settle down." You will be considering the following questions for each of the two methods for convection.

1. How do the "time to settle down" and oscillation amplitude vary as you increase `dt` to 0.125?

2. How do the "time to settle down" and oscillation amplitude vary as you decrease `nu` to `5.e-5`?

3. How do the "time to settle down" and oscillation amplitude vary as you increase the maximum boundary velocity (`Um` to 5.0?

4. How do the "time to settle down" and oscillation amplitude vary if you perform one uniform mesh refinement (`mesh=refine(mesh)`? If you perform two mesh refinements?

It turns out that the variation of "time to settle down" and amplitude when *two* parameters are varied together (*e.g.,* refining the mesh and reducing `nu`) can be very interesting. If, for example, reducing `nu` causes the solution to fail in some way, refining the mesh might improve matters. Feel free to investigate further.

4 Higher order time differencing

The "lagged" time differencing method used in the examples is

$$\begin{aligned}\frac{u^{k+1}-u^k}{\Delta t} - \nu\nabla^2 u^{k+1} + (u^k\cdot\nabla)u^{k+1} + \nabla p^{k+1} &= f^{k+1}\\ \nabla\cdot u^{k+1} &= 0\end{aligned}$$

This method is only first order in time ($O(\Delta t)$). The analogous second order method is based on Crank-Nicolson time differencing and is

$$\begin{aligned}\frac{u^{k+1}-u^k}{\Delta t} - \nu\nabla^2\frac{u^{k+1}+u^k}{2} + ((\frac{3}{2}u^k - \frac{1}{2}u^{k-1})\cdot\nabla)\frac{u^{k+1}+u^k}{2} + \nabla p^{k+1} &= \frac{f^{k+1}+f^k}{2}\\ \nabla\cdot u^{k+1} &= 0\end{aligned}$$

This method is second order in time ($O(\Delta t^2)$) for velocity and also for pressure, if p^{k+1} is interpreted as an approximation of $p(\frac{t^{k+1}+t^k}{2})$. This method shares with the first order case the fact that each time step presents a *linear* problem to solve.

Implement this higher-order method and compare the plotted results of vertical velocity behind the cylinder for the same values of `dt`.