Quick analysis of convergence speed of some time integration (time-marching) methods for steady-state advection-diffusion problem - FEB-MAR'19

Equation that we consider $\frac{\partial c}{\partial t} + u \cdot \frac{\partial c}{\partial x} + v \cdot \frac{\partial c}{\partial y} = \nu \cdot \left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} \right)$.

Introduce dimensionless variables $\sigma_x = \frac{u \cdot \Delta t}{\Delta x}$, $\sigma_y = \frac{v \cdot \Delta t}{\Delta y}$, $cellReinv_x = \frac{\nu}{u \cdot \Delta x}$, $cellReinv_y = \frac{\nu}{v \cdot \Delta y}$, and write equation as

$$\Delta t \cdot \frac{\partial c}{\partial t} + \sigma_x \cdot \Delta x \cdot \frac{\partial c}{\partial x} + \sigma_y \cdot \Delta y \cdot \frac{\partial c}{\partial y} = \sigma_x \cdot cellReinv_x \cdot \Delta x^2 \cdot \frac{\partial^2 c}{\partial x^2} + \sigma_y \cdot cellReinv_y \cdot \Delta y^2 \cdot \frac{\partial^2 c}{\partial y^2} .$$

Write $\sigma_x = \cos\alpha \cdot Cour$, $\sigma_y = \sin\alpha \cdot Cour$, $\sigma_x \cdot cellReinv_x = Cour \cdot cReinv_x$, and $\sigma_y \cdot cellReinv_y = Cour \cdot cReinv_y$, with $\alpha = \arctan\left(\frac{v/\Delta y}{u/\Delta x} \right)$ flow direction,

$Cour = \Delta t \cdot \sqrt{(u/\Delta x)^2 + (v/\Delta y)^2}$ Courant number, and with $cReinv_x = \frac{\nu}{\Delta x^2 \cdot \sqrt{(u/\Delta x)^2 + (v/\Delta y)^2}}$ and $cReinv_y = \frac{\nu}{\Delta y^2 \cdot \sqrt{(u/\Delta x)^2 + (v/\Delta y)^2}}$. Equation becomes $\Delta t \cdot \frac{\partial c}{\partial t} + Cour \cdot \left(\cos\alpha \cdot \Delta x \cdot \frac{\partial c}{\partial x} + \sin\alpha \cdot \Delta y \cdot \frac{\partial c}{\partial y} - cReinv_x \cdot \Delta x^2 \cdot \frac{\partial^2 c}{\partial x^2} - cReinv_y \cdot \Delta y^2 \cdot \frac{\partial^2 c}{\partial y^2} \right) = 0$.

27MAR'19: for analysis of convergence speed of fractional-step method based on subsequent implicit time integration of wave part and convection-viscosity part (effect of splitting is in 2D same as in 1D, hence analysis of 1D case is sufficient), we consider the non-dimensionalized 1D shallow-water equations (cf. "shallowwatertimeint-fracstep.mw"; no need to consider separate v -momentum equation): $\frac{\partial \zeta}{\partial t} + \frac{\partial u}{\partial x} + Fr \cdot \frac{\partial \zeta}{\partial x} = 0$, $\frac{\partial u}{\partial t} + Fr \cdot \frac{\partial u}{\partial x} + \frac{\partial \zeta}{\partial x} = Reinv \cdot \frac{\partial^2 u}{\partial x^2}$.

Non-conservative form for the moment, simplest way to quickly get an idea of fractional-step performance.

For the purpose of the analysis, we write this as $\Delta t \cdot \frac{\partial \zeta}{\partial t} + Cour \cdot \Delta x \cdot \left(\frac{\partial u}{\partial x} + Fr \cdot \frac{\partial \zeta}{\partial x} \right) = 0$, $\Delta t \cdot \frac{\partial u}{\partial t} + Cour \cdot \Delta x \cdot \left(Fr \cdot \frac{\partial u}{\partial x} + \frac{\partial \zeta}{\partial x} \right) = Cour \cdot \Delta x^2 \cdot cReinv \cdot \frac{\partial^2 u}{\partial x^2}$, with $Cour$ the wave Courant number.

```
> restart;
  with(LinearAlgebra):  with( plots ):
```

AF scheme, using 2nd-order accurate central differences: fractional-step implicit time integration.
NB, use of a different θ per direction is possible, but then mismatch between implicit part per direction, so may not be a good idea. => To be investigated.

```
> AFconv := proc( alpha, cReinvx, cReinvy, thetaA, thetaD )   # TWO θ's added to allow evaluation of both fully implicit ( θ = 1 ) and time-accurate ( θ = 1 / 2 ) for Advection and Diffusion separately
          local  aux, g;
```

```
            global  kDx, kDy, Cour;
              aux := ( 1 + Cour * ( thetaA*cos(alpha)*I*sin(kDx) - thetaD*cReinvx*(2*cos(kDx)-2) ) )
                   * ( 1 + Cour * ( thetaA*sin(alpha)*I*sin(kDy) - thetaD*cReinvy*(2*cos(kDy)-2) ) ) * (g - 1)
                     = - Cour * ( cos(alpha)*I*sin(kDx) + sin(alpha)*I*sin(kDy) - cReinvx*(2*cos(kDx)-2) -
cReinvy*(2*cos(kDy)-2) );
              -log10( abs(solve(aux, g)) );
            end proc:
```

```
> # dummy statement to skip next one
```

First AF extension: RBAF. Bad idea (as we realized a bit late), because doesn't compensate for approximate-factorization splitting error.

```
> # continue
```

Second AF extension: sort of RBAF, with compensation for factorization error. Doesn't seem to be a very good idea either, probably because compensation is not perfect (cure is worse than the pain).

```
> # continue
```

27MAR'19: fractional-step method applied to 1D shallow-water equations **discretized on collocated grid** (easy to adapt to staggered grid, but not done yet).

```
> # continue
```

What about implicit solve per direction with intermediate update of right-hand side? => Sort of ADI. => MUCH WORSE THAN AF!
ADI scheme, using 2nd-order accurate central differences: fractional-step implicit time integration.
NB, use of a different θ per direction is possible, but then mismatch between implicit part per direction, so may not be a good idea. => To be investigated.

```
> ADIconv := proc( alpha, cReinvx, cReinvy, theta )   #
```

θ added to allow evaluation of both sort-of-emulated fully implicit ($\theta = 2$) and more-or-less time-accurate ($\theta = 1$)

```
            local  auxx, auxy, gx, gy;
            global  kDx, kDy, Cour;
              auxx := ( 1 + theta * Cour * ( cos(alpha)*I*sin(kDx) - cReinvx*(2*cos(kDx)-2) ) ) * (gx - 1)
                    = - Cour * ( cos(alpha)*I*sin(kDx) + sin(alpha)*I*sin(kDy) - cReinvx*(2*cos(kDx)-2) -
cReinvy*(2*cos(kDy)-2) );
              auxy := ( 1 + theta * Cour * ( sin(alpha)*I*sin(kDy) - cReinvy*(2*cos(kDy)-2) ) ) * (gy - 1)
                    = - Cour * ( cos(alpha)*I*sin(kDx) + sin(alpha)*I*sin(kDy) - cReinvx*(2*cos(kDx)-2) -
cReinvy*(2*cos(kDy)-2) );
              -log10( abs(solve(auxy,gy)*solve(auxx,gx)) );
            end proc:
```

FM scheme, using 1st-order accurate upwind differences (ensure that $\cos\alpha$, $\sin\alpha \geq 0$ => $0 \leq \alpha \leq \pi/2$):explicit time integration.

```
> FMconv := proc( alpha, cReinvx, cReinvy, thetaC )   #
```

θ_C added to allow investigation of effect of a predictor-corrector approach, cf. the Maple worksheet "shallowwatertimeint-fracstep.mw"

```
            local  auxpred, auxcorr;
            global  kDx, kDy, Cour;
              auxpred := 1 - Cour * ( cos(alpha)*(1-exp(I*kDx)) + sin(alpha)*(1-exp(I*kDy)) - cReinvx*(2*cos
```

```
(kDx)-2) - cReinvy*(2*cos(kDy)-2) );
            auxcorr := 1 - Cour * ( cos(alpha)*(1-exp(I*kDx)) + sin(alpha)*(1-exp(I*kDy)) - cReinvx*(2*cos
(kDx)-2) - cReinvy*(2*cos(kDy)-2) )
                           * (thetaC*auxpred + 1 - thetaC);
            -log10( abs(auxcorr) );
          end proc:
```

BE scheme, using 2nd-order accurate central differences: backward Euler fully implicit time integration.

```
> BEconv := proc( alpha, cReinvx, cReinvy, theta )   # θ added to allow evaluation of both fully implicit ( θ = 1 ) and time-accurate ( θ = 1 / 2 )
          local  aux, g;
          global  kDx, kDy, Cour;
            aux := ( 1 + theta * Cour * ( cos(alpha)*I*sin(kDx) - cReinvx*(2*cos(kDx)-2) )
                       + theta * Cour * ( sin(alpha)*I*sin(kDy) - cReinvy*(2*cos(kDy)-2) ) ) * (g - 1)
                   = - Cour * ( cos(alpha)*I*sin(kDx) + sin(alpha)*I*sin(kDy) - cReinvx*(2*cos(kDx)-2) -
cReinvy*(2*cos(kDy)-2) );
            -log10( abs(solve(aux, g)) );
          end proc:
> evalf( BEconv(0,Pi/4,Pi/4,1) );
```

$$-0.4342944819 \ln\left(\frac{2.}{\left|-2.\,\mathrm{I}\,Cour\sin(kDx)+3.141592654\,Cour\cos(kDx)+3.141592654\,Cour\cos(kDy)-6.283185308\,Cour-2.\right|}\right)$$

The plots.

```
> thetaA := 1.0;   # advection θ
  thetaD := 1.0;   # diffusion θ (0.8 is more or less optimal)
  level := 0*0.023;   # just some level (0.023, 0.03)
  dir := evalf(  45 * Pi/180):
  # dir := evalf( -45*0 * Pi/180):
  # icRex := 0.5:  icRey := 0.5:  # icRey := 0.0:
  icRex := 0.1:  icRey := 0.1:
  kDlist := [ seq( evalf(Pi/2^n), n=0..7 ) ];
  aux := unapply( AFconv(dir, icRex, icRey, thetaA, thetaD), [Cour,kDx,kDy] ):
```

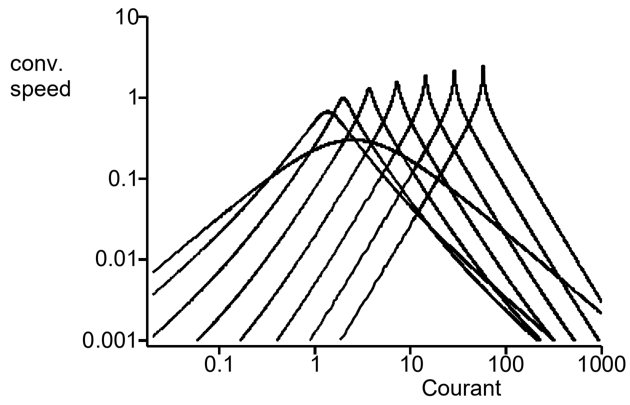
$$thetaA := 1.0$$

$$thetaD := 1.0$$

$$level := 0.$$

$$kDlist := [3.141592654, 1.570796327, 0.7853981635, 0.3926990818, 0.1963495409, 0.09817477044, 0.04908738522, 0.02454369261]$$

```
> loglogplot( [ level, seq( aux(Cour,kD,kD), kD=kDlist ) ], Cour=0.02..1.0E3, 0.001..10, numpoints=10000,
  color=["Black"],
            axis=[thickness=1], axesfont=[Arial,11], labelfont=[Arial,11], size=[400,300], tickmarks=[[seq
  (10^i,i=-1..3)],[seq(10^i,i=-3..1)]],
            labels=["                    Courant","conv.\nspeed\n\n\n\n\n\n\n\n"], linestyle=[2,1,1,
  1,1,1,1,1,1,1,1] );
```

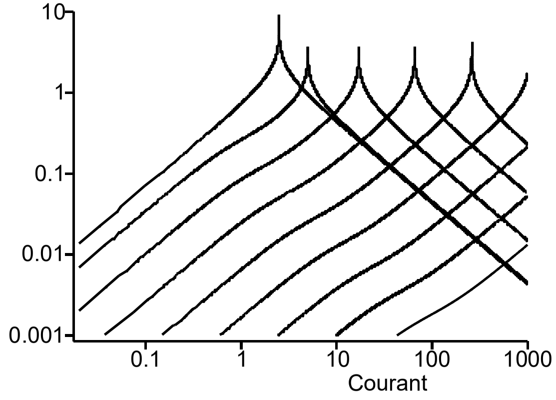



```
> thetaADI := 1.0;   # with θ_ADI = 1  nice general convergence behavior, but less than with AF; with θ_ADI > 1 (1.5 .. 2)  very good to very bad behavior
  dir := evalf(  45 * Pi/180):
  dir := evalf( -45*0 * Pi/180):
  icRex := 0.5:  icRey := 0.5:  # icRey := 0.0:
  icRex := 0.1:  icRey := 0.1:
  kDlist := [ seq( evalf(Pi/2^n), n=0..8 ) ];
  aux := unapply( ADIconv(dir, icRex, icRey, thetaADI), [Cour,kDx,kDy] ):
```

$thetaADI := 1.0$

$kDlist := [3.141592654, 1.570796327, 0.7853981635, 0.3926990818, 0.1963495409, 0.09817477044, 0.04908738522, 0.02454369261, 0.01227184630]$

```
> loglogplot( [ seq( aux(Cour,kD,kD), kD=kDlist ) ], Cour=0.02..1.0E3, 0.001..10, numpoints=10000, color=
  ["Black"],
             axis=[thickness=1], axesfont=[Arial,11], labelfont=[Arial,11], size=[400,300], tickmarks=[[seq
  (10^i,i=-1..3)],[seq(10^i,i=-3..1)]],
             labels=["                              Courant","conv.\nspeed\n\n\n\n\n\n\n\n"] );
```

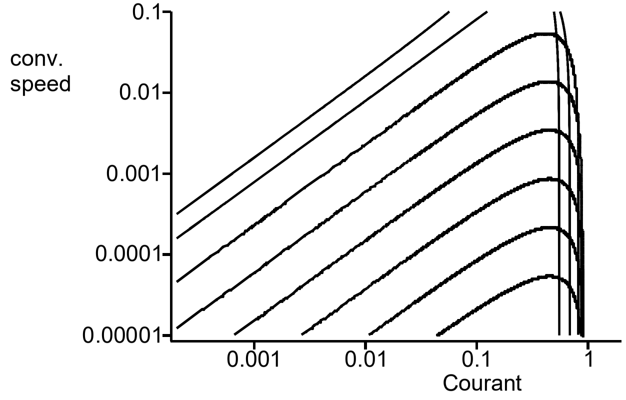



```
> thetaC := 0;   # amount of predictor applied; 0.15 gives close to optimal convergence speed (for the case considered), but only slightly more than factor 2 improvement (for 2 times more work, hm!)
  dir := evalf( 45 * Pi/180):
  icRex := 0.5:  icRey := 0.5:
  icRex := 0.1:  icRey := 0.1:
  kDlist := [ seq( evalf(Pi/2^n), n=0..7 ) ];
  aux := unapply( FMconv(dir, icRex, icRey, thetaC), [Cour,kDx,kDy] ):
```

$$thetaC := 0$$

$$kDlist := [3.141592654, 1.570796327, 0.7853981635, 0.3926990818, 0.1963495409, 0.09817477044, 0.04908738522, 0.02454369261]$$

```
> # loglogplot( [ seq( aux(Cour,kD,kD), kD=kDlist ), 2.7E-5, 6.5E-5 ], Cour=0.02..2.0, 0.00001..0.0001, numpoints=10000, color=["Black"],
  loglogplot( [ seq( aux(Cour,kD,kD), kD=kDlist ) ], Cour=0.0002..2.0, 0.00001..0.1, numpoints=10000, color=["Black"],
            axis=[thickness=1], axesfont=[Arial,11], labelfont=[Arial,11], size=[400,300],
            tickmarks=[[seq(10^i,i=-3..1)],[0.00001=typeset("0.00001"),0.0001=typeset("0.0001"),0.001=typeset("0.001"),0.01=typeset("0.01"),0.1=typeset("0.1")]], # [seq(10^i,i=-6..-1)]],
            labels=["                              Courant","conv.\nspeed\n\n\n\n\n\n\n\n"] );
```



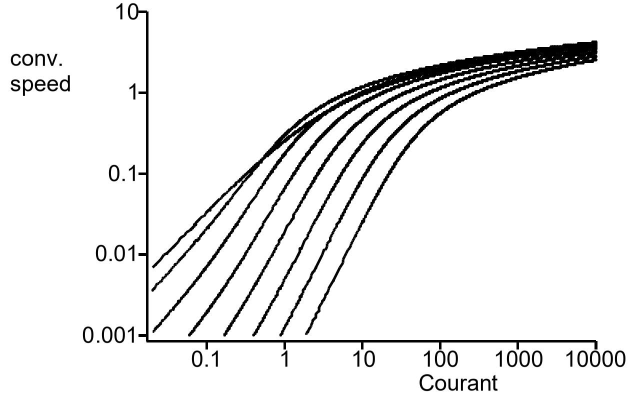

```
> thetaBE := 1;
  dir := evalf( 45 * Pi/180):
  icRex := 0.5:  icRey := 0.5:
  icRex := 0.1:  icRey := 0.1:
  kDlist := [ seq( evalf(Pi/2^n), n=0..7 ) ];
  aux := unapply( BEconv(dir, icRex, icRey, thetaBE), [Cour,kDx,kDy] ):
  coef := 10000*0.1;   # coefficient of dotted line showing consistent time behavior (conv.speed linear in Courant number), mainly to allow easy turn-off of that line
```

$thetaBE := 1$

$kDlist := [3.141592654, 1.570796327, 0.7853981635, 0.3926990818, 0.1963495409, 0.09817477044, 0.04908738522, 0.02454369261]$

$coef := 1000.0$

```
> loglogplot( [ coef*Cour, seq( aux(Cour,kD,kD), kD=kDlist ) ], Cour=0.02..1.0E4, 0.001..10, numpoints=10000, color=["Black"],
            axis=[thickness=1], axesfont=[Arial,11], labelfont=[Arial,11], size=[400,300], tickmarks=[[seq(10^i,i=-1..4)],[seq(10^i,i=-3..1)]],
            labels=["                    Courant","conv.\nspeed\n\n\n\n\n\n\n\n"], linestyle=[2,1,1,1,1,1,1,1,1,1,1] );
```


conv.
speed
10
1
0.1
0.01
0.001
0.1
1
10
100
1000
10000
Courant