Scientific Computing TU Berlin Winter 2021/22 © Jürgen Fuhrmann
Notebook 28

```
begin
using PlutoUI    ,VoronoiFVM    ,HypertextLiteral    ,ExtendableGrids    ,PlutoVista
 ,GridVisualize
default_plotter!(PlutoVista)
end;
```


Convection-diffusion problems

So far, we discussed movement of a chemical species, temperature etc. due to concentration gradients. How can we describe an outer force triggering the motion ? Examples could be species concentration in flowing water, or charged particles moving in an electric field.

Let us start with the stationary case.

Search function $u : \Omega \to \mathbb{R}$ such that

$$\begin{aligned} -\nabla \cdot (D\vec{\nabla} u - u\vec{v}) = f \quad \text{in } \Omega \\ + \text{Boundary conditions} \end{aligned}$$

- $u(x)$: species concentration, temperature...
- $\vec{j} = D\vec{\nabla} u - u\vec{v}$: species flux
- D: diffusion coefficient
- $\vec{v}(x)$: velocity of medium (e.g. a fluid)
- Possible ways to describe $\vec{v}$:
  - Given analytically
  - Solution of free flow problem (Navier-Stokes equation)
  - Flow in porous medium (Darcy equation): $\vec{v} = -\kappa\vec{\nabla} p$ where

$$-\nabla \cdot (\kappa\vec{\nabla} p) = 0$$

- In some important cases, the divergence conditon $\nabla \cdot \vec{v} = 0$ holds.

Finite volume discretization

Let us repeat the finite volume discretization ansatz for Robin boundary conditions:

$$
\begin{aligned}
-\nabla\cdot\vec{j} &= 0 \quad \text{in } \Omega\\
\vec{j}\cdot\vec{n} + \alpha u &= g \quad \text{on } \Gamma = \partial\Omega
\end{aligned}
$$

- Integrate equation over control volume

$$
\begin{aligned}
0 &= -\int_{\omega_k} \nabla\cdot\vec{j}\,d\omega = -\int_{\partial\omega_k} \vec{j}\cdot\vec{n}_k d\gamma\\
&= -\sum_{l\in\mathcal{N}_k}\int_{\sigma_{kl}} \vec{j}\cdot\vec{n}_{kl} d\gamma - \int_{\gamma_k} \vec{j}\cdot\vec{n} d\gamma\\
&\approx \underbrace{\sum_{l\in\mathcal{N}_k}\frac{|\sigma_{kl}|}{h_{kl}} g_{kl}(u_k,u_l)}_{\to A_\Omega} + \underbrace{|\gamma_k|\alpha u_k}_{\to A_\Gamma} - |\gamma_k| g_k
\end{aligned}
$$

- We can split the matrix into a domain part and a boundary part: $A = A_\Omega + A_\Gamma$

- g_{kl} approximates normal convective-diffusive flux between control volumes ω_k, ω_l: $g_{kl}(u_k - u_l) \approx -(D\vec{\nabla}u - u\vec{v})\cdot n_{kl}$
- Let $\sigma_{kl} = \omega_k \cap \omega_l$
- Let $v_{kl} = \frac{1}{|\sigma_{kl}|}\int_{\sigma_{kl}} \vec{v}\cdot\vec{n}_{kl} d\gamma$ approximate the normal velocity $\vec{v}\cdot\vec{n}_{kl}$

Central difference flux

Central difference flux:

$$
\begin{aligned}
g_{kl}(u_k,u_l) &= D(u_k - u_l) + h_{kl}\frac{1}{2}(u_k+u_l)v_{kl}\\
&= (D + \frac{1}{2}h_{kl}v_{kl})u_k - (D - \frac{1}{2}h_{kl}v_{kl})u_l
\end{aligned}
$$

- Evaluation of $u\vec{v}_{kl} \approx \frac{1}{2}(u_k + u_l)$ by averaging u along grid edge – the boundary between two Voronoi cells intersects the grid edge exactly in the center.
- multiplication by h_{kl} comes from the particular scaling of g_{kl} which takes the division by h_{kl} out of the function
- if v_{kl} is large compared to h_{kl}, the corresponding matrix (off-diagonal) entry may become positive
- Non-positive off-diagonal entries only guaranteed for $h \to 0$!

Upwind flux

- If all off-diagonal entries would be non-positive, we can prove the discrete maximum principle and the M-property of the discretization matrix
- Force correct sign of convective flux approximation by replacing central difference flux approximation $h_{kl}\frac{1}{2}(u_k + u_l)v_{kl}$ by

$$
h_{kl}u\vec{v}_{kl} \approx \left(\begin{cases} h_{kl}u_k v_{kl}, & v_{kl} < 0\\ h_{kl}u_l v_{kl}, & v_{kl} > 0\end{cases}\right) = h_{kl}\frac{1}{2}(u_k+u_l)v_{kl} + \underbrace{\frac{1}{2}h_{kl}|v_{kl}|}_{\text{Artificial Diffusion } \tilde{D}}(u_k - u_l)
$$

Upwind flux:

$$
\begin{aligned}
g_{kl}(u_k,u_l) &= D(u_k-u_l) + \begin{cases} h_{kl}u_kv_{kl}, & v_{kl}>0 \\ h_{kl}u_lv_{kl}, & v_{kl}<0 \end{cases} \\
&= (D+\tilde{D})(u_k-u_l) + h_{kl}\frac{1}{2}(u_k+u_l)v_{kl}
\end{aligned}
$$

- M-Property guaranteed unconditonally !
- Artificial diffusion introduces error: second order approximation replaced by first order approximation

Exponential fitting flux

- Project equation onto edge $x_K x_L$ of length $h = h_{kl}$, let $v = -v_{kl}$, integrate once

$$\begin{aligned} u' - uv &= j \\ u|_0 &= u_k \\ u|_h &= u_l \end{aligned}$$

- Linear ODE

Solution of the homogeneus problem:

$$\begin{aligned} u' - uv &= 0 \\ u'/u &= v \\ \ln u &= u_0 + vx \\ u &= K\exp(vx) \end{aligned}$$

Solution of the inhomogeneous problem:

- Set $K = K(x)$:

$$\begin{aligned} K' \exp(vx) + vK\exp(vx) - vK\exp(vx) &= -j \\ K' &= -j\exp(-vx) \\ K &= K_0 + \frac{1}{v} j \exp(-vx) \end{aligned}$$

- Therefore,

$$\begin{aligned} u &= K_0 \exp(vx) + \frac{1}{v} j \\ u_k &= K_0 + \frac{1}{v} j \\ u_l &= K_0 \exp(vh) + \frac{1}{v} j \end{aligned}$$

Insert boundary conditions

$$\begin{aligned} K_0 &= \frac{u_k - u_l}{1 - \exp(vh)} \\ u_k &= \frac{u_k - u_l}{1 - \exp(vh)} + \frac{1}{v} j \\ j &= \frac{v}{\exp(vh) - 1}(u_k - u_l) + vu_k \\ &= v\left(\frac{1}{\exp(vh) - 1} + 1\right)u_k - \frac{v}{\exp(vh) - 1}u_l \\ &= v\left(\frac{\exp(vh)}{\exp(vh) - 1}\right)u_k - \frac{v}{\exp(vh) - 1}u_l \\ &= \frac{-v}{\exp(-vh) - 1}u_k - \frac{v}{\exp(vh) - 1}u_l \\ &= \frac{B(-vh)u_k - B(vh)u_l}{h} \end{aligned}$$

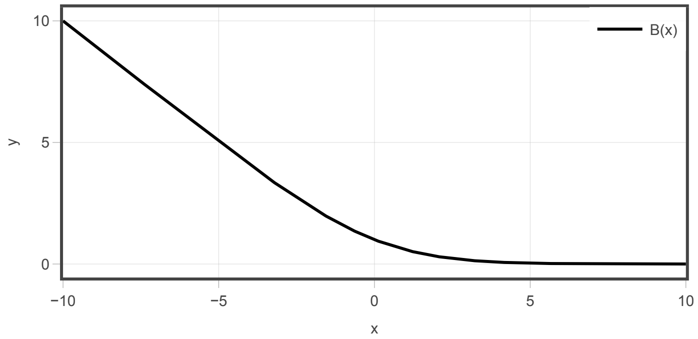
where $B(\xi) = \frac{\xi}{\exp(\xi) - 1}$: Bernoulli function

- General case: $Du' - uv = D(u' - u\frac{v}{D})$

Exponential fitting upwind flux:

$$g_{kl}(u_k, u_l) = D(B(\frac{-v_{kl}h_{kl}}{D})u_k - B(\frac{v_{kl}h_{kl}}{D})u_l)$$

```
B (generic function with 1 method)
B(x)=x≈0.0 ? 1 :  x/(exp(x)-1)
```



- Allen+Southwell 1955
- Scharfetter+Gummel 1969
- Ilin 1969
- Chang+Cooper 1970
- Guaranteed sign pattern, M property!

Artificial diffusion

- Difference of exponential fitting scheme and central scheme
- Use: $B(-x) = B(x) + x \Rightarrow$

$$B(x) + \frac{1}{2}x = B(-x) - \frac{1}{2}x = B(|x|) + \frac{1}{2}|x|$$

$$\begin{aligned} D_{art}(u_k - u_l) =& D(B(\frac{-vh}{D})u_k - B(\frac{vh}{D})u_l) - D(u_k - u_l) + h\frac{1}{2}(u_k + u_l)v \\ =& D(\frac{-vh}{2D} + B(\frac{-vh}{D}))u_k - D(\frac{vh}{2D} + B(\frac{vh}{D})u_l) - D(u_k - u_l) \\ =& D\left(\frac{1}{2}|\frac{vh}{D}| + B(|\frac{vh}{D}|\right) - 1)(u_k - u_l) \end{aligned}$$

- Further, for $x > 0$:

$$\frac{1}{2}x \geq \frac{1}{2}x + B(x) - 1 \geq 0$$

- Therefore

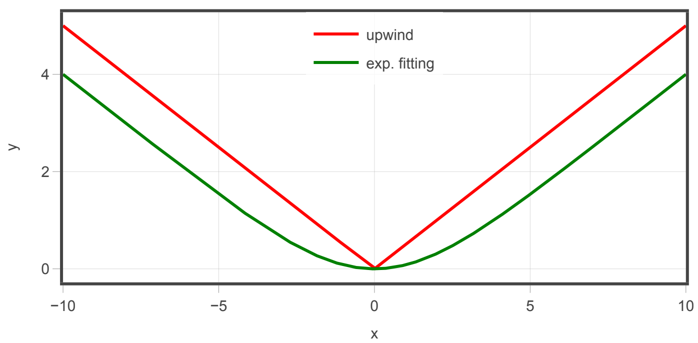
$$\frac{|vh|}{2} \geq D_{art} \geq 0$$

```
dartupw (generic function with 1 method)
```

```
dartupw(vh)=abs(vh)*0.5
```

```
dartexp (generic function with 1 method)
```

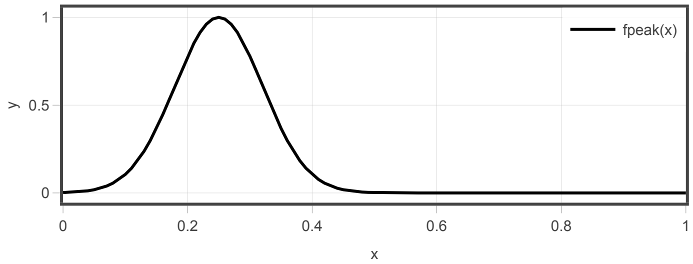
```
dartexp(vh)=abs(vh)/2+B(abs(vh))-1
```



Compared to the simple upwind flux, the artificial diffusion introduced by the exponential fitting scheme is minimized.

Test in VoronoiFVM.jl

$$
\begin{aligned}
\partial_t u - \nabla \cdot (D\vec{\nabla} u - u\vec{v}) &= 0 \quad \text{in } \Omega\\
D\vec{\nabla} u - u\vec{v} \cdot \vec{n} &= 0 \quad \text{on } \partial\Omega\\
u|_{t=0} &= u_0
\end{aligned}
$$




```
fpeak (generic function with 2 methods)
```

```
begin
    fpeak(x)=exp(-100*(x-0.25)^2)
    fpeak(x,y)=exp(-100*((x-0.25)^2+(y-0.25)^2))
end
```

```
create_grid (generic function with 1 method)
```

```
function create_grid(nx,dim)
    X=collect(0:1.0/nx:1)
    if dim==1
      grid=simplexgrid(X)
    else
      grid=simplexgrid(X,X)
    end
end
```

```
convection_diffusion (generic function with 1 method)
```

```julia
function convection_diffusion(;
        n=20,
        dim=1,
        tstep=1.0e-5,
        tend=1,
        dirichlet=true,
        D=0.001,
        vx=10.0,
        vy=10.0,
        scheme="expfit")

    grid=create_grid(n,dim)
    # copy vx, vy into vector
    if dim==1
        V=[vx]
    else
        V=[vx,vy]
    end
    # Bernoulli function


     B(x)=x/(exp(x)-1)

     function flux_expfit!(f,u,edge)
        vh=project(edge,V)  # Calculate projection v * (x_L-x_K)
        f[1]=D*(B(-vh/D)*u[1,1]- B(vh/D)*u[1,2])
     end

     function flux_centered!(f,u,edge)
        vh=project(edge,V)
        f[1]=D*(u[1,1]-u[1,2])+ vh*0.5*(u[1,1]+u[1,2])
     end

     function flux_upwind!(f,u,edge)
        vh=project(edge,V)
        f[1]=D*(u[1,1]-u[1,2])+ ( vh>0.0 ? vh*u[1,1] : vh*u[1,2] )
     end

     flux! =flux_upwind!
     if scheme=="expfit"
        flux! =flux_expfit!
     elseif scheme=="centered"
        flux! =flux_centered!
     end

     ## Storage term (under time derivative)
     function storage!(f,u,node)
        f[1]=u[1]
     end
     function bc!(f,u,node)
        if dirichlet
        boundary_dirichlet!(f,u,node,region=2, value=0)
        boundary_dirichlet!(f,u,node,region=3, value=0)
        end
     end

     sys=VoronoiFVM.System(grid;flux=flux!,storage=storage!, species=
[1],bcondition=bc!)

     ## Create a solution array
     inival=unknowns(sys)

     ## Broadcast the initial value
     inival[1,:].=map(fpeak,grid)

     control=VoronoiFVM.SolverControl()
     control.Δt_min=0.01*tstep
     control.Δt=tstep
     control.Δt_max=0.01*tend
     control.Δu_opt=0.1

     tsol=solve(inival,sys,[0,tend];control=control)
     return grid,tsol
end
```

Conv
Fini
Ce
U
E
Tes

```
100
dim=1; n=100
```

scheme: centered / expfit / upwind dirichlet: ☑

```
grid,tsol=convection_diffusion(;scheme=scheme[1],dim,dirichlet,n);
```

Timestep: 1

Time=0.0

```
sol =
[0.00193045, 0.00315111, 0.00504176, 0.00790705, 0.0121552, 0.0183156, 0.0270518, 0.039163
sol=tsol[1,:,t]
```