第32回 OpenFOAM 勉強会 for beginner @ 関東

OpenFOAM の理論と実装のお話

2013年8月24日

春日 悠

目次

- 対流項の離散化スキームの比較
- 非直交補正
- 運動方程式の拡散項の離散化
- 対流項の有界性確保

対流項の離散化スキームの比較

・中心差分スキーム
linear
cubic

・風上差分スキーム
upwind
linearUpwind
linearUpwind with faceLimited
QUICK
quadraticUpwindFit
cubicUpwindFit

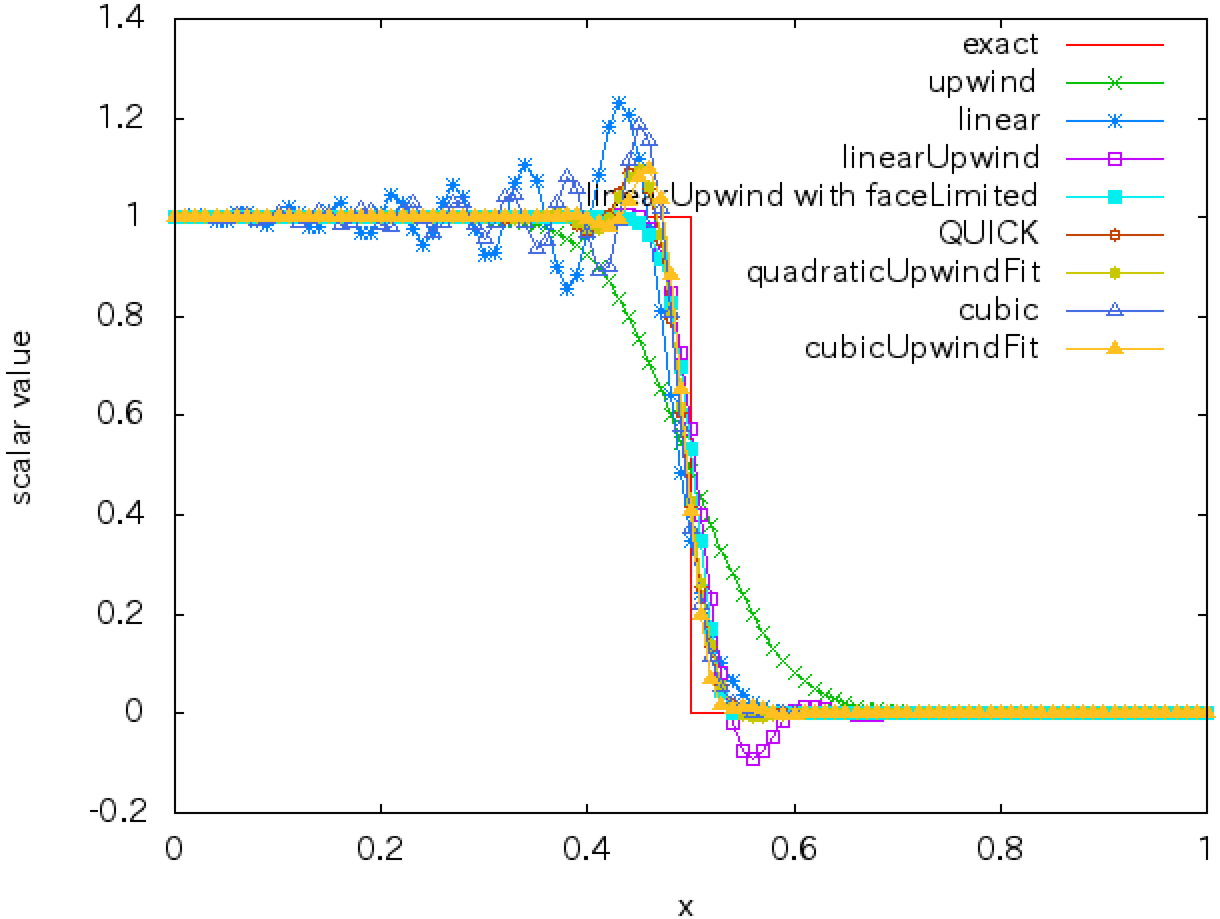
・TVD スキーム
Minmod
SuperBee
vanLeer
vanAlbada
UMIST
MUSCL
limitedLinear
limitedCubic

・NVD スキーム
Gamma

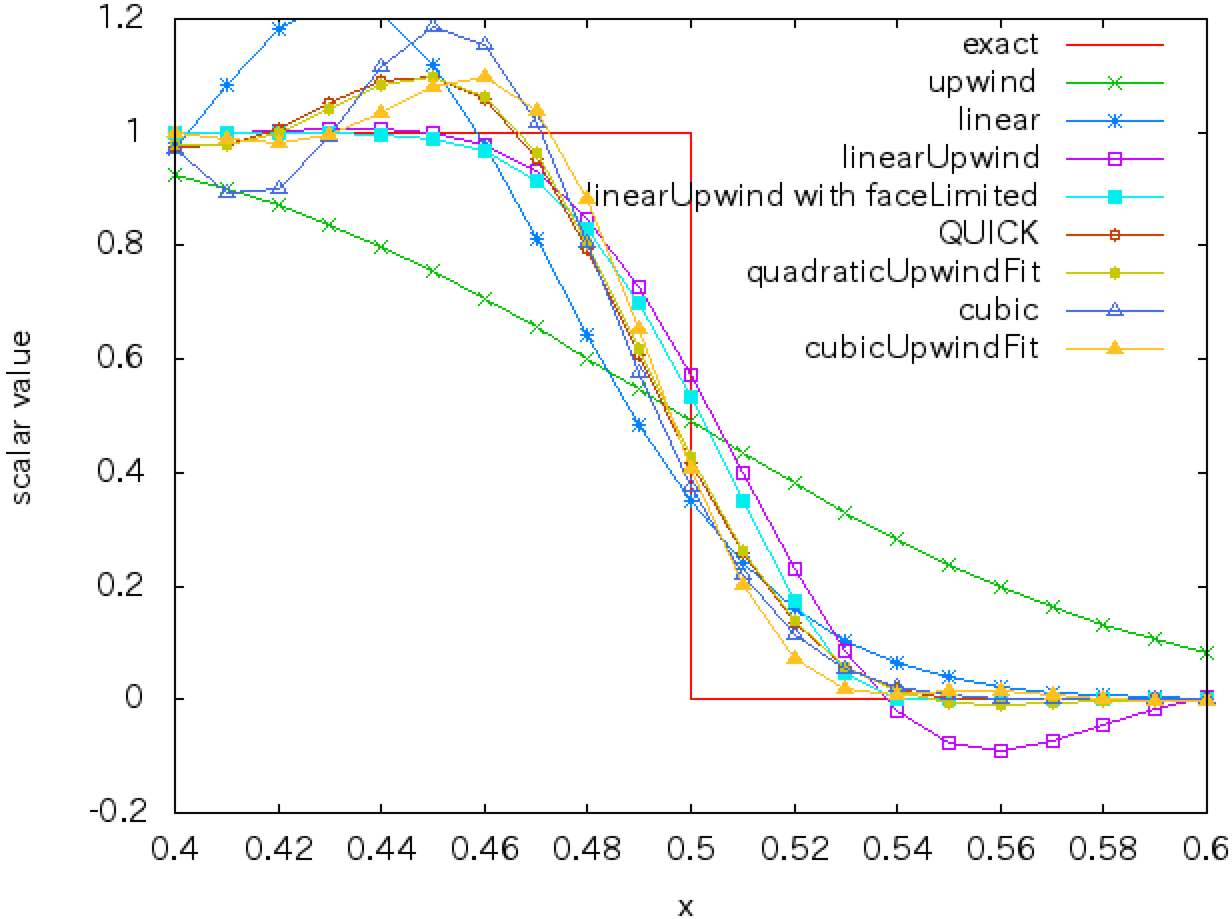
・制限関数適用
QUICK with Minmod
QUICK with SuperBee
QUICK with vanAlbada

スキームの評価ポイント

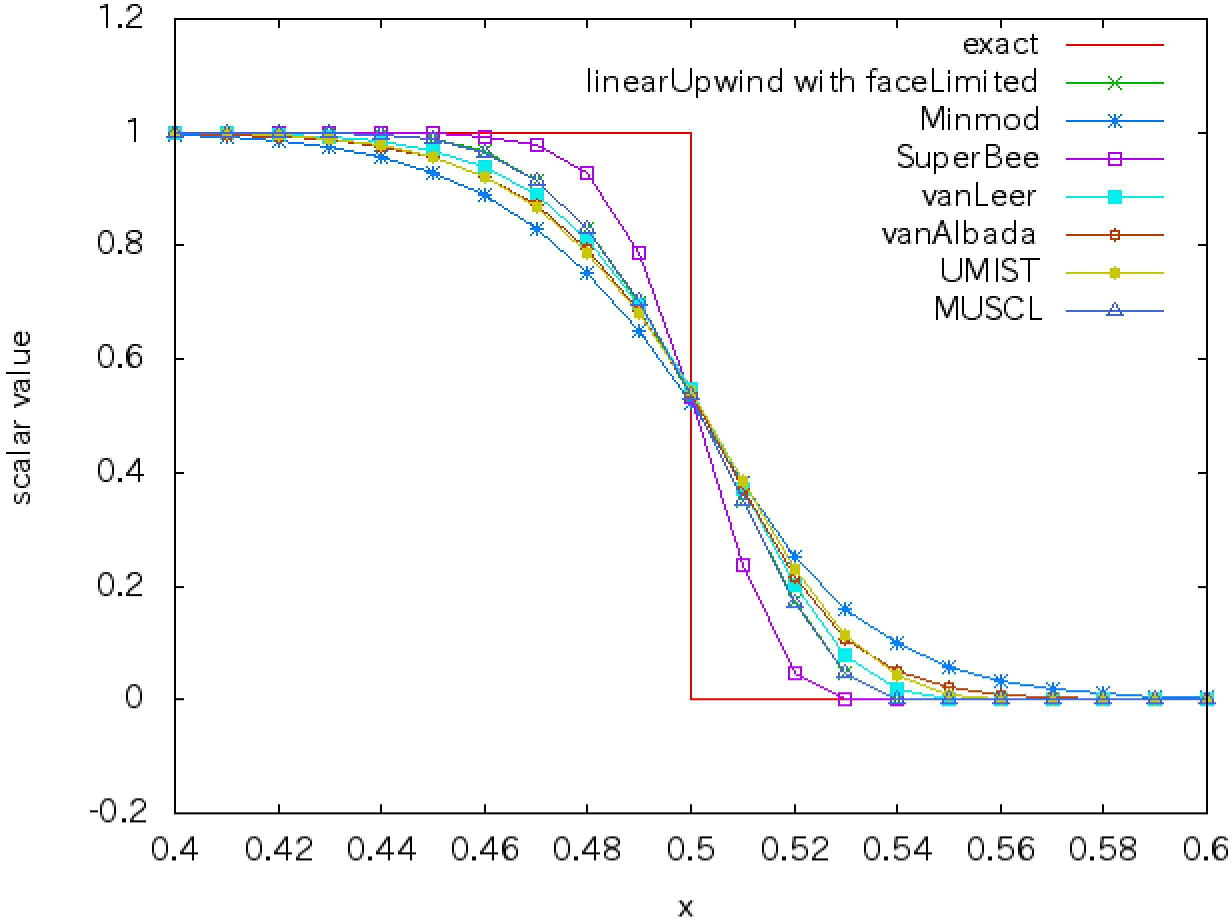
- 安定性と精度のほどよいバランスが重要。
- 数値拡散が増すと精度は落ちるが安定性は増す。ほどよい数値拡散をもつものがよい。
- 単調性 (オーバーシュート・アンダーシュートしない) もポイント。



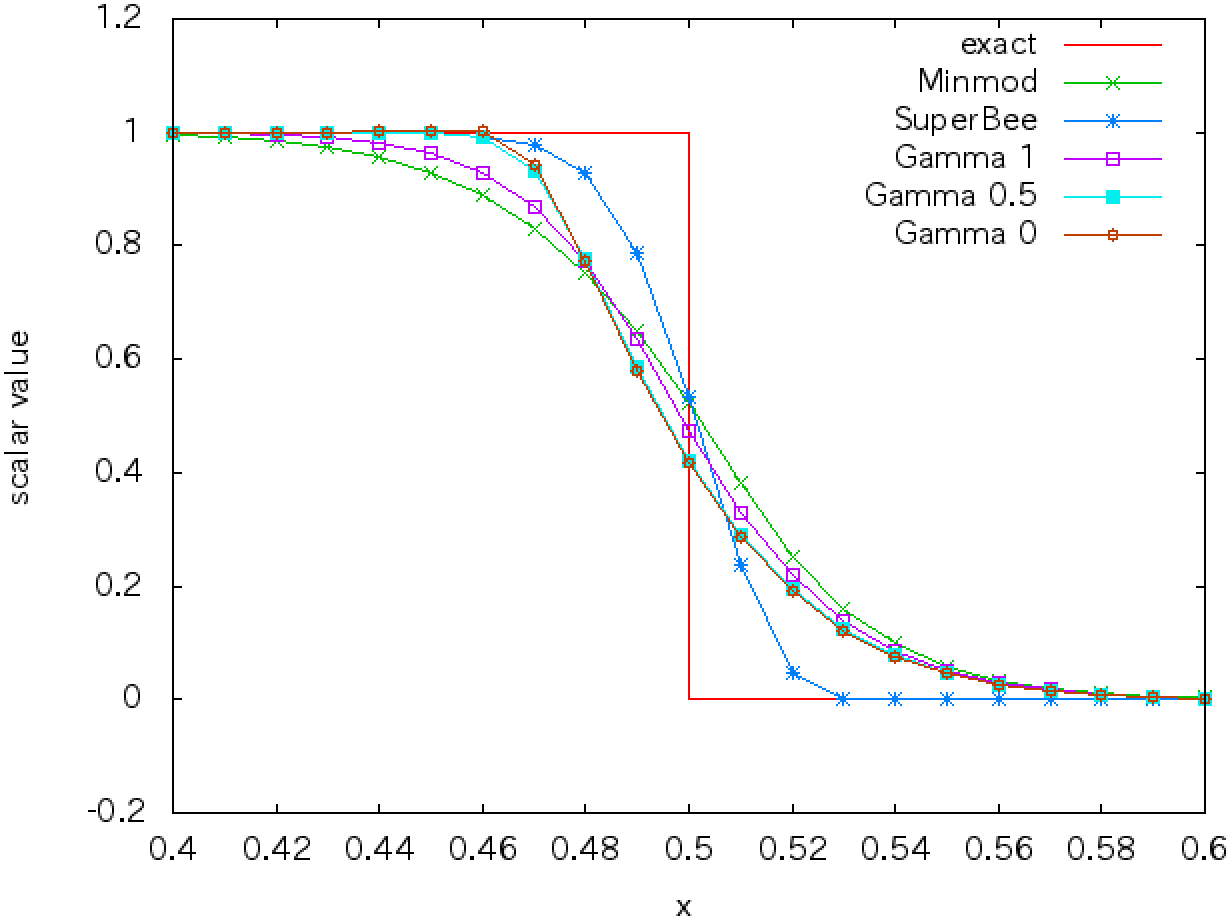
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
scalar value
0
0.2
0.4
0.6
0.8
1
x
exact
upwind
linear
linearUpwind
linearUpwind with faceLimited
QUICK
quadraticUpwindFit
cubic
cubicUpwindFit



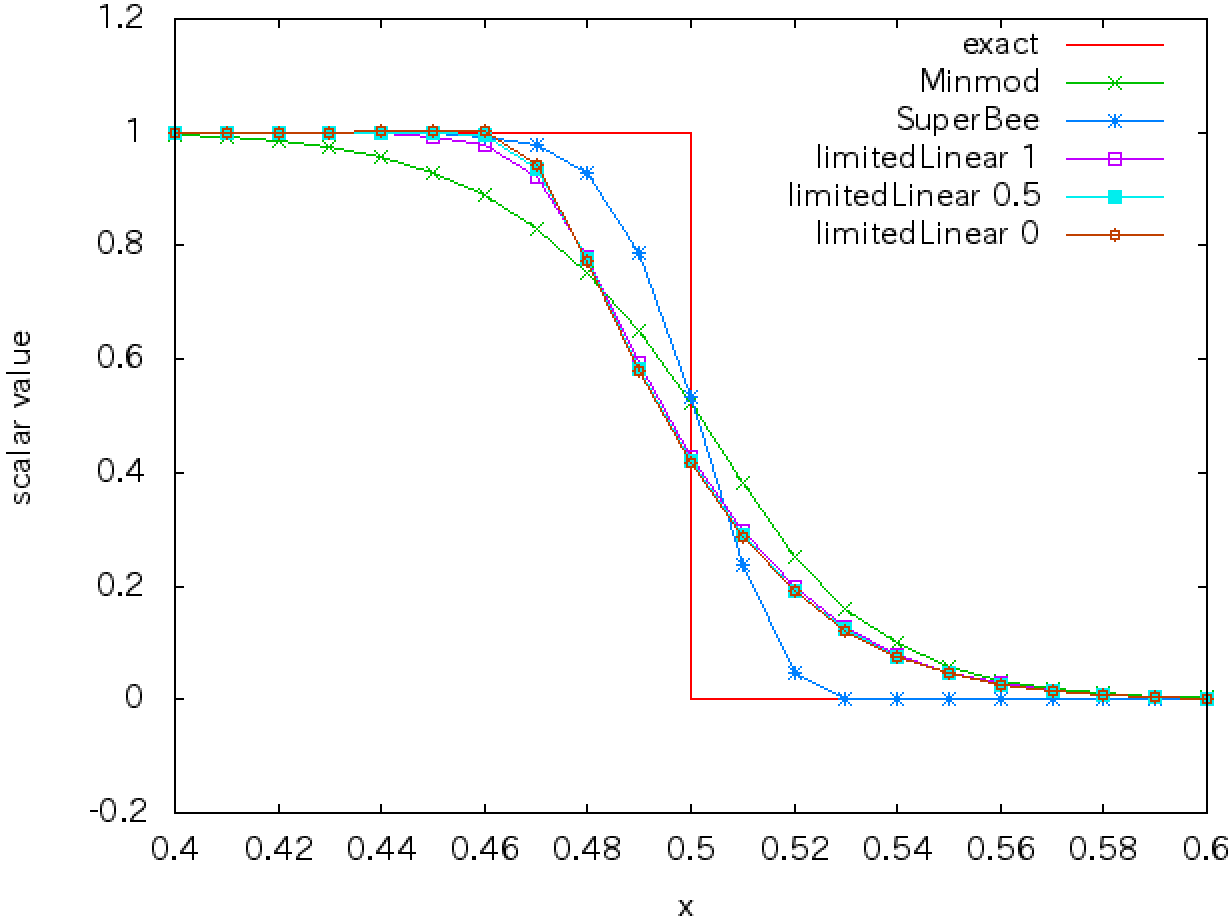
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0.4
0.42
0.44
0.46
0.48
0.5
0.52
0.54
0.56
0.58
0.6
scalar value
x
exact
upwind
linear
linearUpwind
linearUpwind with faceLimited
QUICK
quadraticUpwindFit
cubic
cubicUpwindFit

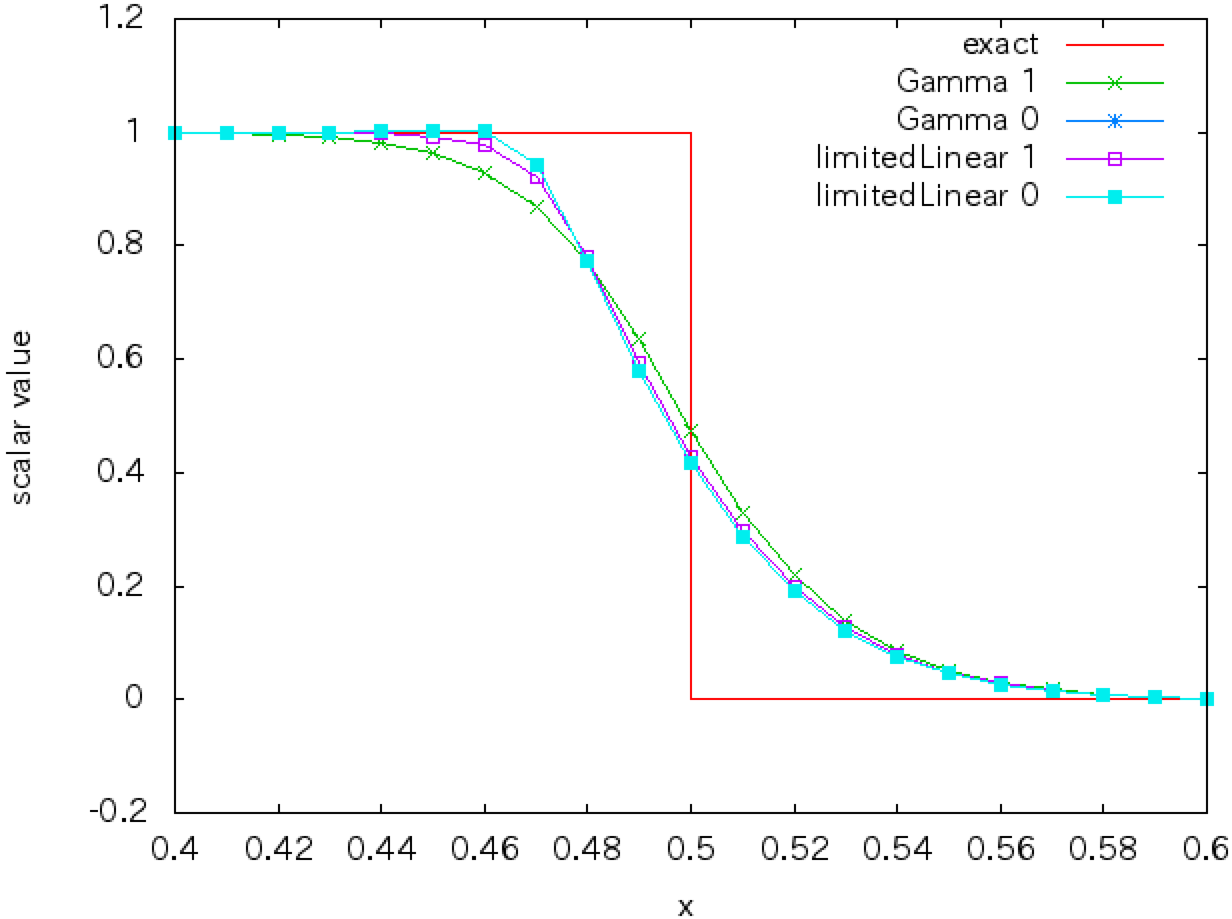


exact
linearUpwind with faceLimited
Minmod
SuperBee
vanLeer
vanAlbada
UMIST
MUSCL
scalar value
x
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0.4
0.42
0.44
0.46
0.48
0.5
0.52
0.54
0.56
0.58
0.6

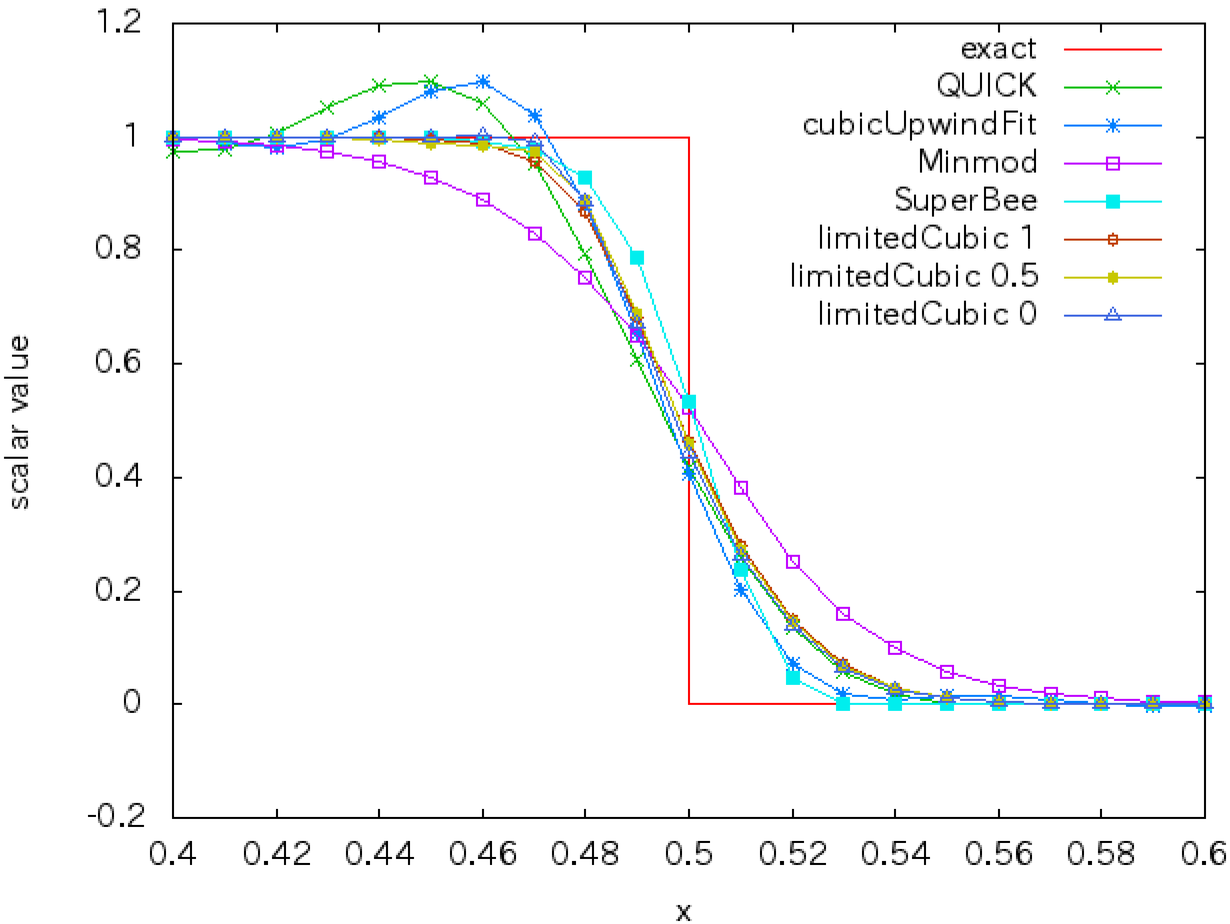


exact
Minmod
SuperBee
Gamma 1
Gamma 0.5
Gamma 0
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
scalar value
0.4
0.42
0.44
0.46
0.48
0.5
0.52
0.54
0.56
0.58
0.6
x

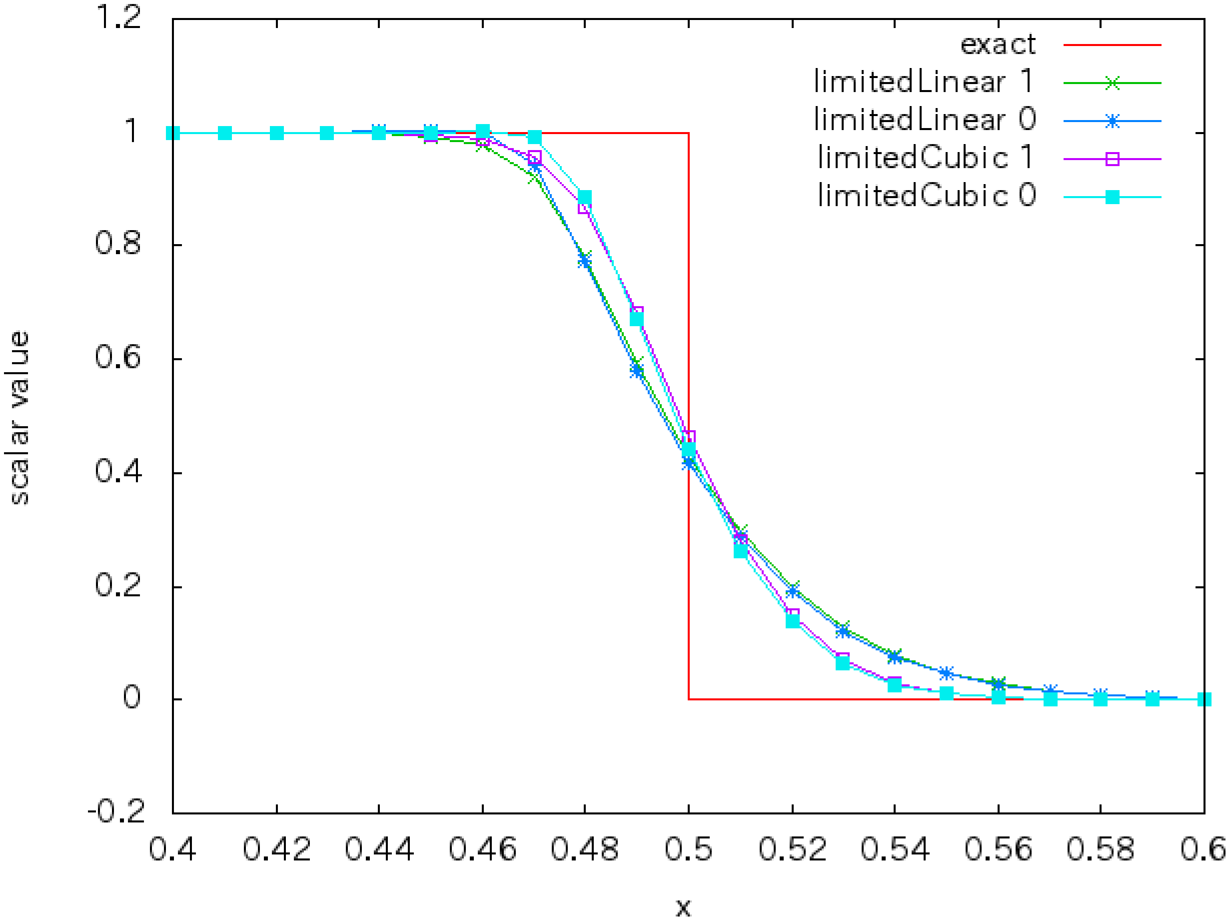

exact
Minmod
SuperBee
limitedLinear 1
limitedLinear 0.5
limitedLinear 0
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
scalar value
0.4
0.42
0.44
0.46
0.48
0.5
0.52
0.54
0.56
0.58
0.6
x



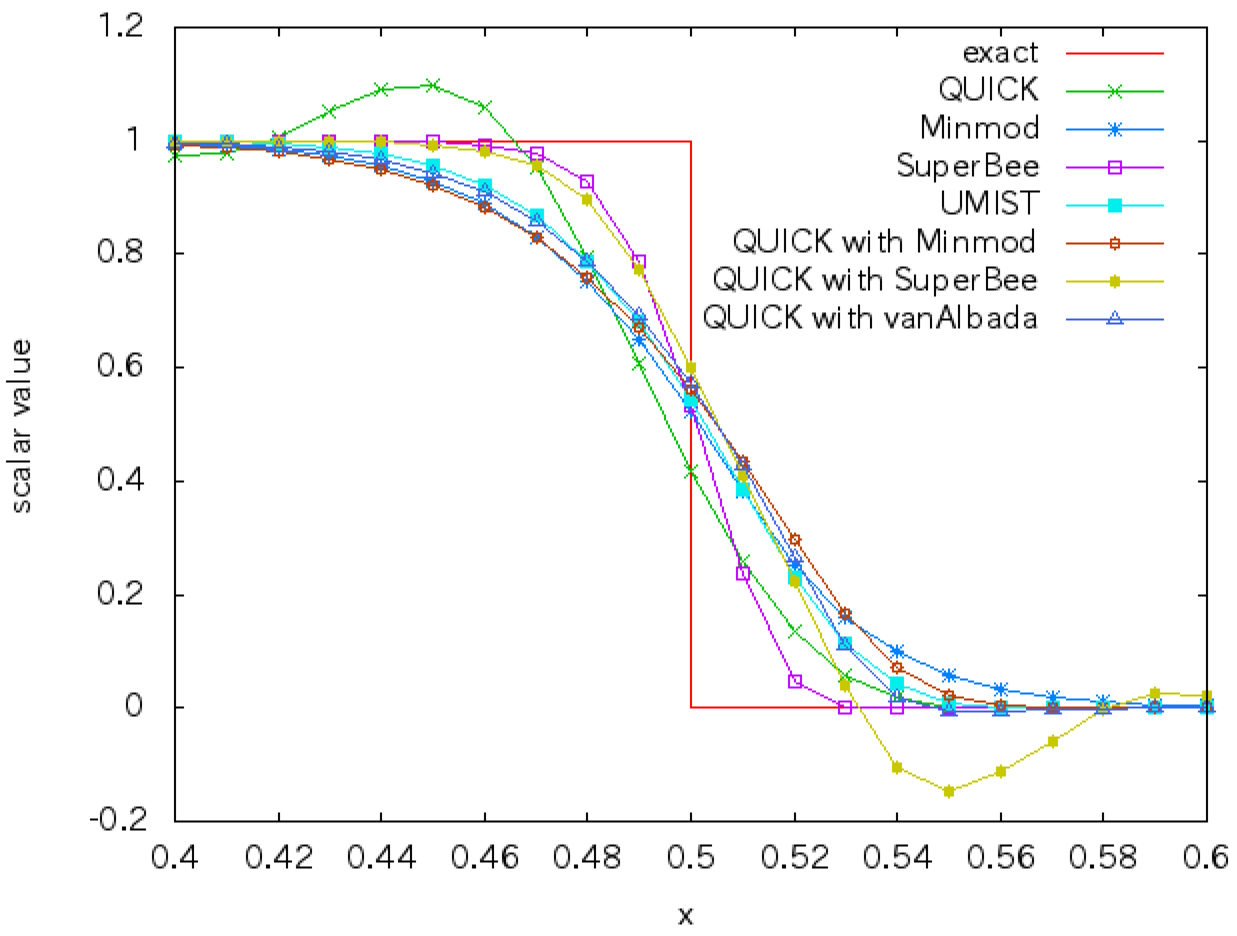
exact
Gamma 1
Gamma 0
limitedLinear 1
limitedLinear 0
scalar value
x
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0.4
0.42
0.44
0.46
0.48
0.5
0.52
0.54
0.56
0.58
0.6



exact
QUICK
cubicUpwindFit
Minmod
SuperBee
limitedCubic 1
limitedCubic 0.5
limitedCubic 0
scalar value
x
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0.4
0.42
0.44
0.46
0.48
0.5
0.52
0.54
0.56
0.58
0.6



exact
limitedLinear 1
limitedLinear 0
limitedCubic 1
limitedCubic 0
scalar value
x
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0.4
0.42
0.44
0.46
0.48
0.5
0.52
0.54
0.56
0.58
0.6


1.2
1
0.8
0.6
0.4
0.2
0
-0.2
scalar value
0.4
0.42
0.44
0.46
0.48
0.5
0.52
0.54
0.56
0.58
0.6
x
exact
QUICK
Minmod
SuperBee
UMIST
QUICK with Minmod
QUICK with SuperBee
QUICK with vanAlbada

考察

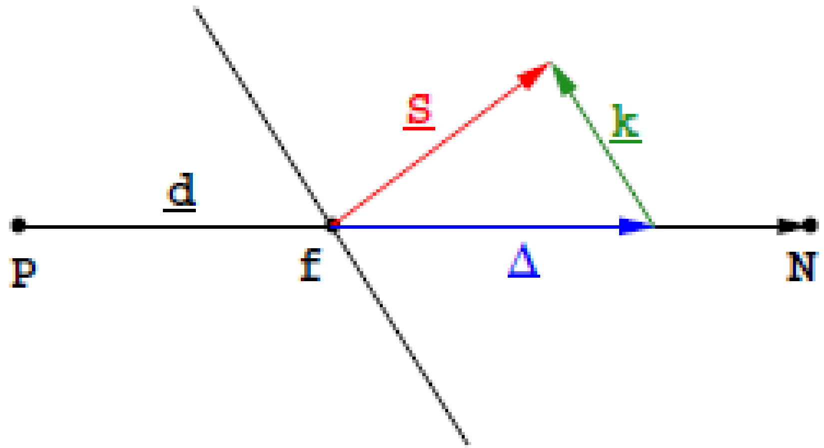
- TVD スキームはどれもたいして変わらないが、同程度の結果を与える勾配制限付き linearUpwind が計算時間の点で有利かもしれない。
- limitedCubic がもっとも精度がよい？ (何次精度？)

非直交補正

- 面の法線方向の勾配 (snGrad) の計算に使われる、セルがゆがんでいるときの補正。
- 直接はもちろんのこと、laplacian でも使われる。

Over-relaxed approach

$$\mathbf{S}.(\nabla\phi)_f = \underbrace{\mathbf{\Delta}.(\nabla\phi)_f}_{\text{orthogonal contribution}} + \underbrace{\mathbf{k}.(\nabla\phi)_f}_{\text{non-orthogonal correction}}.$$



$$\mathbf{\Delta} = \frac{\mathbf{d}}{\mathbf{d}.\mathbf{S}}|\mathbf{S}|^2.$$

※ Jasak 氏論文より

実装

surfaceInterpolation.C

```
vector delta = C[neighbour[facei]] - C[owner[facei]];
vector unitArea = Sf[facei]/magSf[facei];

// Standard cell-centre distance form
//NonOrthDeltaCoeffs[facei] = (unitArea & delta)/magSqr(delta);

// Slightly under-relaxed form
//NonOrthDeltaCoeffs[facei] = 1.0/mag(delta);

// More under-relaxed form
//NonOrthDeltaCoeffs[facei] = 1.0/(mag(unitArea & delta) + VSMALL);

// Stabilised form for bad meshes
nonOrthDeltaCoeffs[facei] = 1.0/max(unitArea & delta, 0.05*mag(delta));
```

d . S があとからかかる

$$\Delta = \frac{\mathbf{d}}{\mathbf{d}.\mathbf{S}}|\mathbf{S}|^2.$$

実装

gaussLaplacianScheme.C

laplacianScheme.C

fvcLaplacian

```
fvcLaplacian(tinterpGammaScheme_().interpolate(gamma)(), vf)
```

```
const surfaceVectorField Sn(mesh.Sf()/mesh.magSf());
const surfaceVectorField SfGamma(mesh.Sf() & gamma);
const GeometricField<scalar, fvsPatchField, surfaceMesh> SfGammaSn
(
    SfGamma & Sn
);
const surfaceVectorField SfGammaCorr(SfGamma - SfGammaSn*Sn);

tmp<GeometricField<Type, fvPatchField, volMesh> > tLaplacian
(
    fvc::div
    (
        SfGammaSn*this->tsnGradScheme_().snGrad(vf)
      + gammaSnGradCorr(SfGammaCorr, vf)
    )
);
```

これは何をしている？

gammaSnGradCorr

```
tgammaSnGradCorr().replace
(
    cmpt,
    SfGammaCorr & fvc::interpolate(fvc::grad(vf.component(cmpt)))
);
```

運動方程式の拡散項

simpleFoam

```
tmp<fvVectorMatrix> UEqn
(
    fvm::div(phi, U)
  + turbulence->divDevReff(U)
  ==
    fvOptions(U)
);
```

rhoSimpleFoam

```
tmp<fvVectorMatrix> UEqn
(
    fvm::div(phi, U)
  + turbulence->divDevRhoReff(U)
 ==
    fvOptions(rho, U)
);
```

運動方程式の拡散項

incompressible/RAS/laminar.C

```
tmp<fvVectorMatrix> laminar::divDevReff(volVectorField& U) const
{
    return
    (
      - fvm::laplacian(nuEff(), U)
      - fvc::div(nuEff()*dev(T(fvc::grad(U))))
    );
}
```

compressible/RAS/laminar.C

```
tmp<fvVectorMatrix> laminar::divDevRhoReff(volVectorField& U) const
{
    return
    (
      - fvm::laplacian(muEff(), U)
      - fvc::div(muEff()*dev2(T(fvc::grad(U))))
    );
}
```

運動方程式の拡散項

TensorI.H

```
//- Return the deviatoric part of a tensor
template<class Cmpt>
inline Tensor<Cmpt> dev(const Tensor<Cmpt>& t)
{
    return t - SphericalTensor<Cmpt>::oneThirdI*tr(t);
}


//- Return the deviatoric part of a tensor
template<class Cmpt>
inline Tensor<Cmpt> dev2(const Tensor<Cmpt>& t)
{
    return t - SphericalTensor<Cmpt>::twoThirdsI*tr(t);
}
```

係数 1/3

係数 2/3

・最後の項は速度の発散で、非圧縮性だと連続の式で消えるが、残してある。
・どうしてこの表現か？ (エレガントさの問題？) 粘性が一定なら係数 1/3 になるが、一定でない (乱流粘性が入る) ときもこうなる？

対流項の有界性確保

```
div(phi,U)      bounded Gauss linearUpwindV grad(U);
```

↑

これ

$$\frac{Dh}{Dt} = \frac{\partial h}{\partial t} + \mathbf{U}\bullet\nabla h = \frac{\partial h}{\partial t} + \nabla\bullet(\mathbf{U}h) - (\nabla\bullet\mathbf{U})h$$

$$\frac{D(\rho h)}{Dt} = \frac{\partial(\rho h)}{\partial t} + \rho\mathbf{U}\bullet\nabla h - \left(\frac{\partial\rho}{\partial t} + \nabla\bullet(\rho\mathbf{U})\right)h$$

最後の項はほんらいいらない子だが、行列の優対角性確保のために残す？

おわりです