

Improving the $k - \varepsilon$ turbulence model for simulation of atmospheric boundary layer flow

Jonathon Sumner

PhD Candidate

Prof. Christian Masson

Canada Research Chair

École de technologie supérieure

June 2, 2009

- 1 Context
- 2 A simple case: Flow under homogeneous conditions
 - Overview
- 3 Alternate FVM discretization schemes
- 4 OpenFOAM implementation
 - Source term corrections
 - Surface normal gradient schemes
 - Interpolation schemes
- 5 Results
- 6 Conclusions

Doctoral research



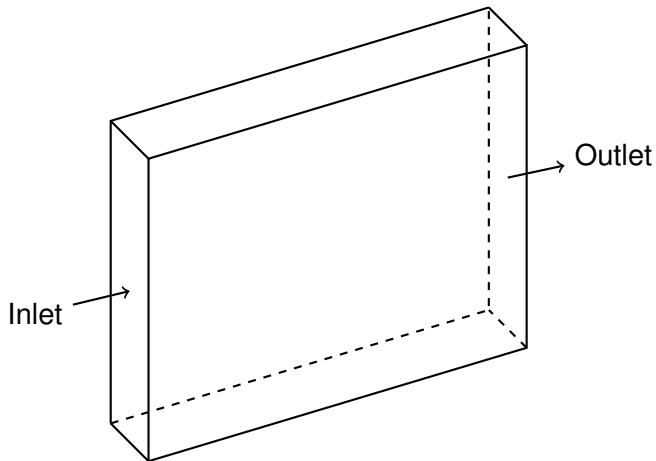
Source: 3Ci Energie

- 1 Context
- 2 A simple case: Flow under homogeneous conditions
 - Overview
- 3 Alternate FVM discretization schemes
- 4 OpenFOAM implementation
 - Source term corrections
 - Surface normal gradient schemes
 - Interpolation schemes
- 5 Results
- 6 Conclusions

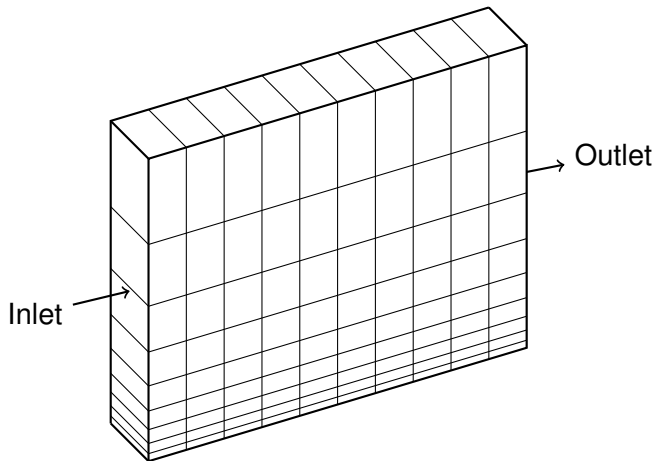
Simulation of neutral ABL flow over homogeneous terrain

- 1 Motivation
- 2 Richards and Hoxey (1993) - Appropriate boundary conditions
- 3 Richards et al (2002) - Blind-test
- 4 Hargreaves and Wright (2007) - Adaptation of commercial software

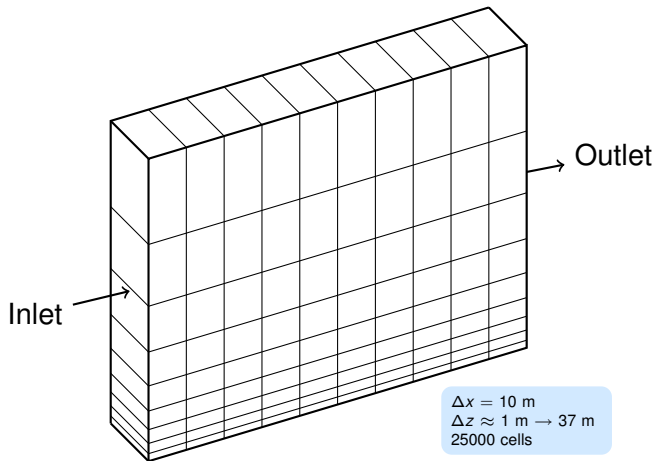
Domain, grid and boundary conditions



Domain, grid and boundary conditions



Domain, grid and boundary conditions

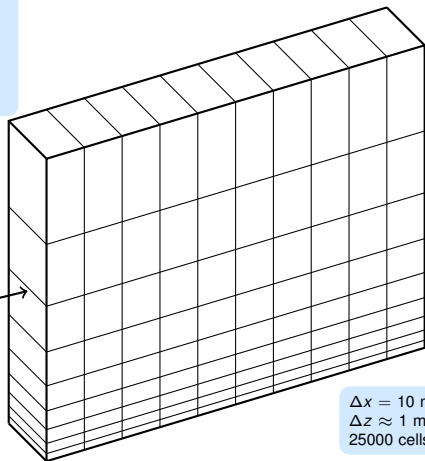


Domain, grid and boundary conditions

$$\tau = \rho u_{abl}^{*2}$$

$$\frac{\partial k}{\partial z} = 0$$

$$\varepsilon = \frac{u_{abl}^{*3}}{\kappa(H + z_0)}$$



Outlet

Zero gradients

Inlet

$$U(z) = \frac{u_{abl}^*}{\kappa} \ln \left(\frac{z + z_0}{z_0} \right)$$

$$k(z) = \frac{u_{abl}^{*2}}{\sqrt{C_\mu}}$$

$$\varepsilon(z) = \frac{u_{abl}^{*3}}{\kappa(z + z_0)}$$

$\Delta x = 10 \text{ m}$
 $\Delta z \approx 1 \text{ m} \rightarrow 37 \text{ m}$
 25000 cells

The lower boundary

Turbulence properties

- Production rate for near-wall cell

$$G_{k,p} = \frac{u_g^{*3}}{\kappa(z_p + z_0)}$$

- Dissipation rate for near-wall cell

$$\varepsilon_p = \frac{\sqrt{C_\mu} k_p u_g^*}{\kappa(z_p + z_0)}$$

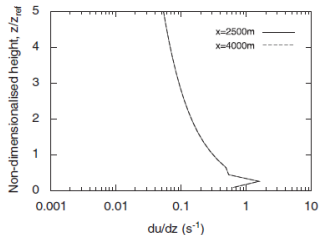
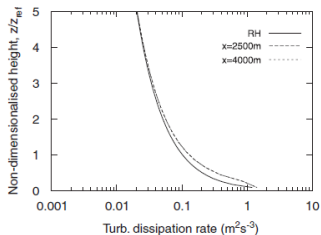
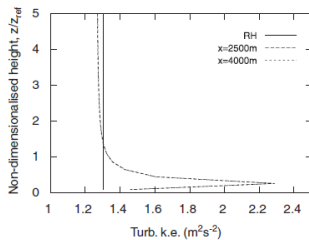
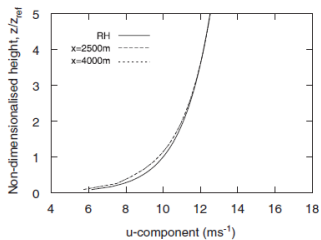
The lower boundary

Velocity

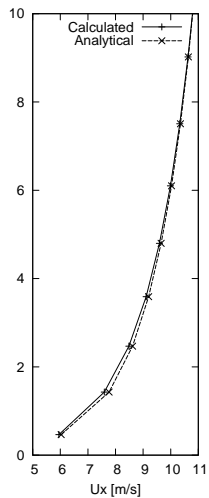
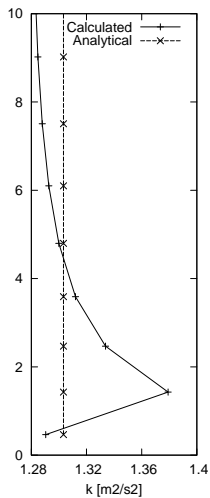
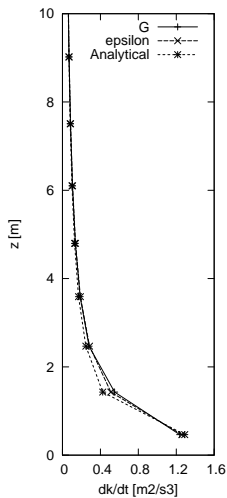
- To ensure proper surface shear stress

$$U_w = U_p - \frac{u_g^{*3}}{\sqrt{C_\mu} k_p \kappa} \frac{z_p}{z_p + z_0}$$

A small problem

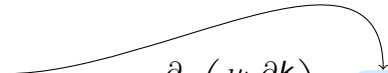


A small problem



Sources of error

- Usually attributed to overestimation of turbulence production rate near wall

$$\frac{\partial}{\partial z} \left(\frac{\nu_t}{\sigma_k} \frac{\partial k}{\partial z} \right) + G_k - \varepsilon = 0$$


$$\frac{\partial}{\partial z} \left(\frac{\nu_t}{\sigma_\varepsilon} \frac{\partial \varepsilon}{\partial z} \right) + C_{\varepsilon 1} G_k \frac{\varepsilon}{k} - C_{\varepsilon 2} \frac{\varepsilon^2}{k} = 0$$

Sources of error

- Usually attributed to overestimation of turbulence production rate near wall
- Piecewise linear approximation of highly non-linear distributions likely root cause

$$\frac{\partial}{\partial z} \left(\frac{\nu_t}{\sigma_k} \frac{\partial k}{\partial z} \right) + G_k - \varepsilon = 0$$

$$\frac{\partial}{\partial z} \left(\frac{\nu_t}{\sigma_\varepsilon} \frac{\partial \varepsilon}{\partial z} \right) + C_{\varepsilon 1} G_k \frac{\varepsilon}{k} - C_{\varepsilon 2} \frac{\varepsilon^2}{k} = 0$$

- 1 Context
- 2 A simple case: Flow under homogeneous conditions
 - Overview
- 3 Alternate FVM discretization schemes**
- 4 OpenFOAM implementation
 - Source term corrections
 - Surface normal gradient schemes
 - Interpolation schemes
- 5 Results
- 6 Conclusions

Turbulence transport equations

2D steady $k - \varepsilon$ model

$$\frac{\partial}{\partial z} \left(\frac{\nu_t}{\sigma_k} \frac{\partial k}{\partial z} \right) + G_k - \varepsilon = 0$$

$$\frac{\partial}{\partial z} \left(\frac{\nu_t}{\sigma_\varepsilon} \frac{\partial \varepsilon}{\partial z} \right) + C_{\varepsilon 1} G_k \frac{\varepsilon}{k} - C_{\varepsilon 2} \frac{\varepsilon^2}{k} = 0$$

where

$$G_k = \nu_t \left(\frac{\partial U}{\partial z} \right)^2$$

Source term corrections

Evaluation of source terms

$$\int_{CV} S dV = \bar{S} \Delta V$$
$$\approx S_p \Delta V$$

Correction factor

$$f = \frac{\bar{S}}{S_p}$$

Source term corrections

Turbulence production rate

- By definition

$$\overline{G_k} = \frac{1}{\Delta V} \int_{CV} \nu_t \left(\frac{\partial U}{\partial z} \right)^2 dV$$

- Cell-centered value

$$G_{k,p} = \nu_{t,p} \left(\frac{\partial U}{\partial z} \right)^2$$

Source term corrections

Correction factors

- For G_k

$$f_G = \frac{\Delta z}{(z_p + z_0)} \cdot \frac{1}{\ln\left(\frac{z_{uf} + z_0}{z_{lf} + z_0}\right)}$$

- For ε

$$f_{\varepsilon 2} = \frac{1}{f_G}$$

Laplacian term corrections

Evaluation of Laplacian terms

$$\begin{aligned}\int_{CV} \nabla \cdot (\gamma \nabla \phi) dV &= \oint_S (\gamma \nabla \phi) \cdot \hat{n} dS \\ &\approx \sum_f \gamma_f (\nabla \phi)_f \cdot \vec{S}_f\end{aligned}$$

Correction factor

$$f = \frac{(\partial \phi / \partial z)|_f}{\Delta \phi / \Delta z}$$

Laplacian term corrections

Surface normal gradient of ε

- Analytical solution indicates

$$\left. \frac{\partial \varepsilon}{\partial z} \right|_f \propto -\frac{1}{z_f^2}$$

- Piecewise linear approximation yields

$$\left. \frac{\partial \varepsilon}{\partial z} \right|_f \approx \frac{\Delta \varepsilon}{\Delta z} \propto -\frac{1}{z_n z_p}$$

Laplacian term corrections

Surface normal gradient correction factors

- For ε

$$f_{\varepsilon 1} = \frac{z_n z_p}{z_f^2}$$

- For U

$$f_U = \frac{z_n - z_p}{z_f \ln \left(\frac{z_n}{z_p} \right)}$$

- Note: if ∇U is explicitly evaluated an interpolation scheme will be used that has the same effect as f_U

Updated transport equations

2D steady $k - \varepsilon$ model

$$\frac{\partial}{\partial z} \left(\frac{\nu_t}{\sigma_k} \frac{\partial k}{\partial z} \right) + \frac{G_k}{f_{\varepsilon 2}} - f_{\varepsilon 2} \varepsilon = 0$$

$$\frac{\partial}{\partial z} \left(\frac{\nu_t}{\sigma_\varepsilon} \frac{\partial \varepsilon}{\partial z} \right) + C_{\varepsilon 1} G_k \frac{\varepsilon}{k} - f_{\varepsilon 2}^2 C_{\varepsilon 2} \frac{\varepsilon^2}{k} = 0$$

- Note: f is purely a function of grid geometry

- 1 Context
- 2 A simple case: Flow under homogeneous conditions
 - Overview
- 3 Alternate FVM discretization schemes
- 4 **OpenFOAM implementation**
 - Source term corrections
 - Surface normal gradient schemes
 - Interpolation schemes
- 5 Results
- 6 Conclusions

kEpsilonRH

```
tmp<fvScalarMatrix> epsEqn
(
    fvm::ddt(epsilon_)
  + fvm::div(phi_, epsilon_)
  - fvm::Sp(fvc::div(phi_), epsilon_)
  - fvm::laplacian(DepsilonEff(), epsilon_)
  ==
    C1_*f1*f2*G_*epsilon_/k_
  - fvm::Sp(C2_*f2*f2*epsilon_/k_, epsilon_)
);
```

kEpsilonRH

```
tmp<fvScalarMatrix> kEqn
(
    fvm::ddt(k_)
    + fvm::div(phi_, k_)
    - fvm::Sp(fvc::div(phi_), k_)
    - fvm::laplacian(DkEff(), k_)
    ==
    f1*G_
    - fvm::Sp(f2*epsilon_/k_, k_)
);
```

Modification of `deltaCoeffs` for ε

```
f3SnGrad<Type>::deltaCoeffs{
    ...
    scalar f3 = 0.0;
    scalar zF = mesh.Cf()[facei].z();
    scalar zP = mesh.C()[owner[facei]].z();
    scalar zN = mesh.C()[neighbour[facei]].z();

    if ((Foam::mag(zP-zN)>SMALL)&&(neighbour[facei]>-1))
        f3 = zP * zN / Foam::sqr(zF);
    else
        f3 = 1.0;

    modDeltaCoeffs[facei] = f3 * deltaCoeffs_[facei];
}
```

Modification of `deltaCoeffs` for U

```
f4SnGrad<Type>::deltaCoeffs{
    ...
    scalar f4 = 0.0;
    scalar zF = mesh.Cf()[facei].z();
    scalar zP = mesh.C()[owner[facei]].z();
    scalar zN = mesh.C()[neighbour[facei]].z();

    if ((Foam::mag(zP-zN)>SMALL)&&(neighbour[facei]>-1))
        f4 = (zN - zP) / (zF * Foam::log(zN / zP));
    else
        f4 = 1.0;

    modDeltaCoeffs[facei] = f4 * deltaCoeffs_[facei];
}
```

Logarithmic interpolation for velocity

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

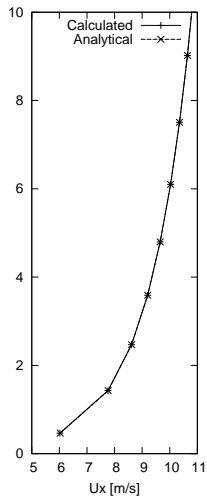
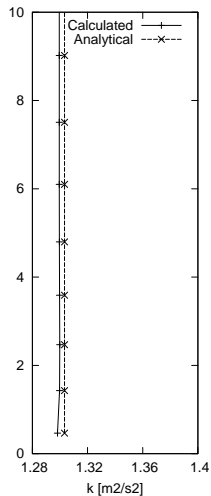
```
tmp<fvVectorMatrix> kEpsilonRH::divDevReff(volVectorField& U)
const {
return
(
    - fvm::laplacian(nuEff(), U)
    - fvc::div(nuEff()*dev(fvc::grad(U)().T()))
);
}
```

Interpolation scheme for `fvc::grad(U)`

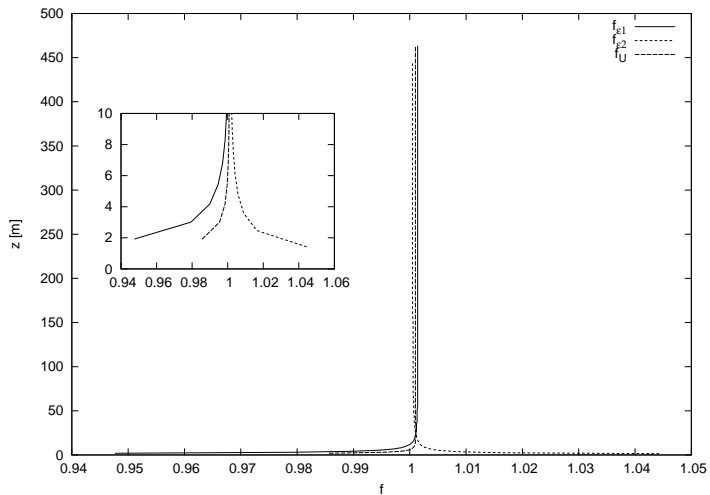
```
logLinear::weights{
    ...
    if ((Foam::mag(zP - zN)>1E-3)&&(neighbour[facei]>-1))
    {
        logLinearWeights[facei] =
            Foam::log(zN/zF) / Foam::log(zN/zP);
    }
    else
    {
        logLinearWeights[facei] =
            Foam::mag(mesh.C()[neighbour[facei]]-mesh.Cf()[facei]) /
            Foam::mag(mesh.C()[neighbour[facei]]-mesh.C()[owner[facei]])
    }
}
```

- 1 Context
- 2 A simple case: Flow under homogeneous conditions
 - Overview
- 3 Alternate FVM discretization schemes
- 4 OpenFOAM implementation
 - Source term corrections
 - Surface normal gradient schemes
 - Interpolation schemes
- 5 **Results**
- 6 Conclusions

Solution with proposed FVM schemes



Wall-function analogy



- 1 Context
- 2 A simple case: Flow under homogeneous conditions
 - Overview
- 3 Alternate FVM discretization schemes
- 4 OpenFOAM implementation
 - Source term corrections
 - Surface normal gradient schemes
 - Interpolation schemes
- 5 Results
- 6 **Conclusions**

An achievement?

A step back

- 1 Resolved a small but annoying numerical error by calibrating numerical schemes with **known** solution

An achievement?

A step back

- 1 Resolved a small but annoying numerical error by calibrating numerical schemes with **known** solution
- 2 Are these corrections valid for non-homogeneous case?

Future work

- 1 From 2D structured orthogonal grid to 3D structured curvilinear grid (to account for irregular lower boundaries).
- 2 Parallelization of new schemes
- 3 Evaluation of effect of numerical schemes on flow solution for non-homogeneous conditions

Thank you

- Questions?