Extra Scalar Transport Equations Capability in SU2

Arturo Cajal

PhD Student

Aerodynamic Design and Research Laboratory

Adviser: A. Uranga

USC University of Southern California

1st SU2 Conference

June 11, 2020

Research Goal

Improve accuracy of RANS for low Reynolds and transitional flows

- Prediction of transition for (wide) range of flows
- Physics-based transition equation (one-point closure)
  - Addition of a transport type scalar equation (under development)
  - Use directly with existing RANS solvers (or with minimal modifications)

Code development

Current transition prediction capability in SU2 is limited to

- Spalart Allmaras - Bas Cakmakcioglu model:
  modification to standard SA model equation
- Langtry-Menter model (2 RANS equations + 2 transition equations): partly coded but non-functional

Our goal is to use a separate equation for transition:

Flow Solver + Turbulence Solver + Transition Solver

- SU2 does not currently have the capability to accept an arbitrary scalar transport equation to be solved alongside the main flow and turbulent solver
- Expand framework to accept extra scalar transport equation(s)

Transition Model

Medina's Laminar Kinetic Energy (LKE) Model

Turbulence model

$$\frac{Dk}{Dt} = \gamma\, f_\nu\, P_k - \gamma\, C_\mu\, k\, \omega + \frac{\partial}{\partial x_j}\left[\left(\nu + \sigma_k\, \gamma\, \frac{k}{\omega}\right)\frac{\partial k_L}{\partial x_j}\right]$$

$$\frac{D\omega}{Dt} = C_{\omega_1}\, P_k\, \frac{\omega}{k} - C_{\omega_2}\, \omega^2 + \frac{\partial}{\partial x_j}\left[\left(\nu + \sigma_\omega\, \gamma\, \frac{k}{\omega}\right)\frac{\partial k_L}{\partial x_j}\right] + \frac{\sigma_d}{\omega}\,\frac{\partial k}{\partial x_j}\,\frac{\partial \omega}{\partial x_j}$$

Transition

$$\frac{Dk_L}{Dt} = P_{k_L} - \epsilon + \frac{\partial}{\partial x_j}\left[\left(\nu + \sigma_{k_L}\, \alpha_L\right)\frac{\partial k_L}{\partial x_j}\right]$$

RANS coupling via $\nu_t = \nu_{t,s} + \nu_L$

where $\nu_{t,s} = f_{ss}\frac{k}{\omega}$, $\nu_L = \dfrac{P_{k_L}}{\max\left(S^2, \left(\frac{||U_i||}{y}\right)^2\right)}$

Medina, H. et al. (2018) A novel laminar kinetic energy model for the prediction of pretransitional velocity fluctuations and boundary layer transition. International Journal of Heat and Fluid Flow. [Online] 69150–163.

Implementation

Convective, viscous, and source terms

Turbulent kinetic energy k

$$\mathbf{F}_k^c = k\,\mathbf{v} \quad , \quad \mathbf{F}_k^\nu = \left(\nu + \sigma_k\,\gamma\,\frac{k}{\omega}\right)\frac{\partial k_L}{\partial x_j} \quad , \quad \mathbf{Q}_k = \gamma\,f_\nu\,P_k - \gamma\,C_\mu\,k\,\omega$$

Specific dissipation ω

$$\mathbf{F}_\omega^c = \omega\,\mathbf{v} \quad , \quad \mathbf{F}_\omega^\nu = \left(\nu + \sigma_\omega\,\gamma\,\frac{k}{\omega}\right)\frac{\partial k_L}{\partial x_j} \quad , \quad \mathbf{Q}_\omega = \gamma\,f_\nu\,P_k - \gamma\,C_\mu\,k\,\omega$$

Laminar kinetic energy $\mathbf{k_L}$

$$\mathbf{F}_{k_L}^c = k_L\,\mathbf{v} \quad , \quad \mathbf{F}_{k_L}^\nu = \left(\nu + \sigma_{k_L}\,\alpha_L\right)\frac{\partial k_L}{\partial x_j} \quad , \quad \mathbf{Q}_{k_L} = P - \epsilon$$

Implementation

`CSolver`

- State vector size definition and allocation
- Loops that control the computation of each term
- Boundary conditions

`CVariable`

- State vector at each mesh node
- Auxiliary data

`CNumerics`

- Numerical schemes for each term
- Existing implementations can be repurposed

Implementation – Solvers 1/3

`CSolvers: CTurbKOmegaLKESolver, CTransLKESolver`

(turbulence solver) *(transition solver)*

Constructor

- State vector dimension
- Define and allocate vectors and structures (single grid for now)
- Initialize solution to far-field state everywhere
- Allocation of inlets

```
CTurbKOmegaLKESolver(CGeometry *geometry, CConfig *
    config)
```

Implementation - Solvers 2/3

CSolvers: CTurbKOmegaLKESolver, CTransLKESolver

Preprocessing & Postprocessing

- Initialize residual vector and Jacobian matrices
- Compute and set turbulent eddy viscosity

```
for (iPoint = 0; iPoint < nPoint; iPoint ++) {
  /*--- Get k and w ---*/
  kine  = nodes->GetSolution(iPoint,0);
  omega = nodes->GetSolution(iPoint,1);
  /*--- Get the laminar eddy viscosity ---*/
  muL = solver_container[TRANS_SOL]->GetNodes()->GetmuL(
      iPoint);
  /*--- Compute the small scale eddy viscosity ---*/
  nuT_small = f_ss * kine / (omega);
  muT_small = nuT_small * rho;
  /*--- Compute the eddy viscosity ---*/
  muT = muT_small + muL;
  muT = max (muT,0.0);
  nodes->SetmuT(iPoint,muT);
 }
```

Implementation – Solvers 3/3

`CSolvers: CTurbKOmegaLKESolver, CTransLKESolver`

Loops for computation of each term

- Upwind and viscous residual methods inhered from `CTurbSolver`
- Source Residual

Boundary Conditions

- Heat flux wall (non-slip)
- Far-field
- Inlet
- Outlet
- Euler and Symmetry boundary conditions inhered from `CTurbSolver`

Implementation – Variables

`CVariable:  CTurbKOmegaLKEVariable, CTransLKEVariable`

(turbulence vars) *(transition vars)*

- Store the solution vector and coupling variables at each mesh node
  e.g., k, ω, eddy viscosity
- Store auxiliary functions: transition initiation function that triggers transition in the LKE model

Implementation – Numerics

`CVariable: CTurbKOmegaLKEVariable, CTransLKEVariable`

(turbulence vars) *(transition vars)*

Convective fluxes: upwind scheme

- Velocity at faces inhered from CUpwScalar method.

Viscous fluxes: cell average gradient method

- Mean gradient approximation inhered from CAvgGradScalar method.

Source term

- Integrates the source terms at each mesh node.

Implementation – Other Considerations

`CDriver.cpp`

- Input and output preprocess
- Solver preprocessing
- Numerics preprocessing
- Integration preprocessing
- Iteration preprocessing

`CFluidIteration::Iterate`

- Include options to solve the transition model using a single zone iteration method within a RANS solver.

Challenges

Implementing a new equation into the SU2 framework *seems* trivial, but there are some challenges to do it:

- Object-oriented "abstraction"
- Inability to visualize local residuals in the domain to debug coding/modeling errors
- Solver coupling
- Numerical divergence (avoid non-physical values)

QUESTIONS / DISCUSSION

Contact: Arturo Cajal, *PhD student*
Aerospace & Mechanical Engineering
University of Southern California

cajal@usc.edu