Development of Physics-Based Transition Models for Unstructured-Mesh CFD Codes Using Deep Learning Models

Chau-Lyan Chang[1]
NASA Langley Research Center, Hampton, VA 23681

Predicting transition locations over a vehicle surface is of fundamental importance for many engineering applications. With the transition information, the Reynolds-averaged Navier-Stokes (RANS) computations can turn on the turbulence model at the right locations so that drag, lift and other aerodynamic quantities can be accurately predicted. In contrast to the popularity of RANS-based transition modeling in which transition onset is governed by the turbulence equations, physics-based transition models that account for instability waves within the boundary layer, thus more compliant to flow physics, only gained more attention in recent years. This paper describes the development of a new physics-based transition model based on either the linear stability theory (LST) or parabolized stability equations (PSE). The model is designed to communicate with a structured or unstructured-mesh RANS solver back and forth in order to more accurately compute transition fronts over a three-dimensional body. In the developed model, the Python suite of interface codes in conjunction with the LASTRAC software can be executed autonomously to produce transition onset locations for a given laminar or RANS-computed transitional state. In addition, as a proof of concept, the tool set consists of a deep learning neural network model that has been designed and trained to predict instability wave evolutions inside the boundary layer for various instability wave mechanisms across a selected speed range. A machine-learned intelligent profile interpolation model has also been devised to enable reliable instability-wave spectra predictions with just a few points in the meanflow profiles.

Nomenclature

A	=	activation function
f	=	disturbance frequency (Hz)
L	=	length of the prolate sphere
l	=	boundary-layer similarity length scale ($=\sqrt{\nu_e x/U_e}$)
M_e	=	boundary-edge Mach number
N	=	integrated disturbance growth factor
N_{CF}	=	crossflow disturbance N factor
N_{TS}	=	Tollmien-Schlichting (TS) wave N factor
$\vec{n}$	=	wall-normal unit vector
P_i	=	the ith surface vertex point
p_r	=	static pressure
Re_x	=	length Reynolds number
Re_θ	=	momentum thickness Reynolds number
R	=	Reynolds number based on boundary-layer length scale ($=\sqrt{Re_x}$)
T_w	=	wall temperature
T_{adw}	=	adiabatic wall temperature
U_e	=	boundary-layer edge velocity
w	=	activation function input

[1] Aerospace Technologist, Computational AeroSciences Branch, Email: Chau-Lyan.Chang@nasa.gov, Associate Fellow, AIAA.

x	=	streamwise coordinate
y	=	wall-normal coordinate
z	=	spanwise coordinate
ρ	=	density
σ	=	disturbance growth rate ($\sigma = -\alpha_i$ for LST)
ϕ	=	flow dependent variable
ϕ'	=	flow dependent variable perturbation
$\bar{\phi}$	=	flow dependent variable mean
$\hat{\phi}$	=	perturbation shape function
Ψ	=	meridian angle of the prolate sphere
ν_e	=	boundary-layer edge kinematic viscosity
ω	=	disturbance temporal wave number
α	=	streamwise wave number
α_i	=	imaginary part of the streamwise wave number
β	=	spanwise wave number

I. Introduction

Laminar to turbulent transition significantly impacts the overall aerodynamic performance of flight vehicles across the speed regimes. Such transition takes place when the near-wall disturbances originated from the environment and surface grow to a large enough magnitude that they can sustain a turbulent boundary layer. Turbulent boundary layers on the surface increase the drag and surface heating substantially and must be accounted for during the design phase of a vehicle development. Modern computational fluid dynamics (CFD) codes in general solve a set of Reynolds-averaged Navier-Stokes (RANS) equations to handle turbulent boundary layers. Transitional boundary layers are often conveniently modeled by the same turbulence-model equations with some modifications (e.g., Refs. [1-3]). These RANS-based transition models, despite their simplifying assumptions regarding the modeling of turbulent quantities in the laminar transition region via a turbulent transport equation, offer a less tedious way to directly compute transitional flows with relatively minor changes of the Navier-Stokes CFD codes. Physics-based transition prediction, on the other hand, is much more involved. A laminar flow solution with an adequate grid resolution is first carried out. Then, instability waves inside the boundary layer are computed with a transition prediction software such as the popular LASTRAC code [4]. Transition fronts are thus determined based on the integrated growth in terms of the N factor, which measures the disturbance amplitude ratio relative to that at the onset of instability waves. For any flow dependent variable perturbed about its laminar mean state such as $\phi = \bar{\phi} + \phi'$, the disturbance can be expressed as

$$\phi'(x,y,z,t) = \hat{\phi}(x,y)e^{i(\beta z-\omega t+\int_{x_0}^{x}\alpha d\xi)} \qquad (1)$$

here $\hat{\phi}, \alpha, \beta,$ and ω are shape function, streamwise, spanwise, and temporal wave numbers, respectively. For spatial instability, the streamwise wave number can be used to integrate the overall disturbance growth as

$$N(x) = -\int_{x_0}^{x}\alpha_i d\xi$$

in which, the growth rate of the disturbance is integrated starting from the location where the flow first becomes unstable. Transition onset is usually determined by an empirically determined N value or values when multiple instability mechanisms are present in the flow. In Eq. (1), if the flow is assumed to be locally parallel, the streamwise dependence of the streamwise wave number and shape functions can be further relaxed and it results in an eigenvalue problem as in the linear stability theory (LST) approach. On the other hand, the approximated parabolized form of partial differential equations derived from the perturbation form given in Eq. (1) are solved in the parabolized stability (PSE) approach. After the disturbance growth factor is obtained from the stability equations, transition onset information is then fed into the RANS computation to properly turn on or off the turbulence model. In addition to the stringent grid requirement, such an approach often requires more expert knowledge and several mean flow/stability computation iterations. Unlike the RANS-based models, it is more difficult to couple the transition prediction procedures to an existing RANS code in this approach. Nevertheless, physics-based transition models provide a means to much more accurately predict transitional flows due to their compliance to flow physics. From the perspective of developing and improving transition prediction capabilities in modern CFD codes as suggested in the CFD Vision 2030 Study [5], incorporating physics-based transition models to existing RANS solvers is in urgent need.

To improve the efficiency and further allow an easier coupling procedure in physics-based transition models, the instability waves evolution step is usually replaced by a computationally more efficient model. In other words, instability wave evolutions are precalculated by using transition prediction software. Disturbance growth rate information is then polynomial-fitted with respect to gross boundary-layer parameters such as momentum thickness Reynolds number (Re_θ) and shape factors for later use by a table lookup procedure [6-7]. Alternatively, due to recent advances in machine learning methodologies, the polynomial fitting and table lookup procedure have been successfully replaced by a pretrained neural network model for a narrowly scoped application [8]. A machine learning model in lieu of the table lookup procedure provides several advantages. First, instead of a gross boundary-layer parameter such as shape factor or Re_θ, the entire flow profiles can be used as the input to the model. Second, a model free of any polynomial form implies that a much broader data range can be fitted into the model. Third, it is possible to combine different instability mechanisms such as Tollmien-Schlichting (TS) waves or crossflow instability into a single model. It is likely that all polynomial-based models would be replaced by more advanced machine learning models in the near future.

In both the RANS-based and table-lookup approaches, transition prediction is carried out by utilizing the existing RANS solver infrastructure, i.e., additional transport equations are being solved for unknowns such as eddy viscosity, amplification factor (N factor) or their surrogates in order to obtain transition information. The main differences among various methods lie in the way the unknowns are modeled. The physics-based transition prediction methods, on the other hand, do not need the RANS infrastructure. Instantaneous laminar flows are usually linearly perturbed to form a set of perturbation equations. These disturbance equations are either solved as an eigenvalue problem as in LST or as a set of approximated parabolic partial differential equations as in the PSE approach. The corresponding instability wave solvers have a very different grid resolution requirement from the CFD solver. Utilizing the existing RANS infrastructure for the LST/PSE solver would greatly limit the prediction accuracy. Transition predictions using LASTRAC are typically done by interpolating given basic (laminar) flows onto body-fitted curvilinear grids for stability computations. Only the boundary-layer profiles are of interest in most cases. Typically, a couple of hundreds grid points with clustering both near the wall and the critical layer in the middle or near the edge of the boundary layer are used. In a three-dimensional (3D) boundary layer, stability computations are often performed along a curved streamline with points that do not coincide with the CFD grids. For these reasons, a pure physics-based transition model using tools such as LASTRAC would require a code infrastructure that is distinctively different from an existing CFD code.

The main objective of this research is to investigate numerical algorithms required to implement an efficient physics-based transition model that can be easily interfaced with an existing structured or unstructured-mesh RANS-CFD solver. The ultimate goal is to have a universal interface suite of codes that can provide on-the-fly transition predictions in existing CFD codes with very minimal user inputs or even hands-free. Towards this goal, four important aspects need to be thoroughly addressed: 1) interfaces with existing codes that allow communications of surface mesh, instantaneous solutions, and transition fronts, 2) intelligent interpolation tools that can accurately interpolate an underresolved (often near the leading edge or at small Reynolds numbers, Re_x) boundary layer to adequate resolutions that a model can use, 3) a machine-learning model that is pretrained to predict instability characteristics with high-enough accuracy such that an N-factor correlation can be applied to determine the transition fronts on the surface, and 4) automatically generated integration paths on any 3D surfaces that allow efficient N factor integrations and markings of transition fronts. In the following sections, a discussion on the development of deep-learning instability wave model is given first, followed by the description of the intelligent interpolation models, and the design and implementation of the CFD codes interface.

II. Deep-Learning Model for Instability Wave Prediction

Deep learning or convolutional neural networks to-date have been predominantly used for image recognitions, pattern recognitions, natural language parser, and classifications. In applying such machine learning models to numerical data predictions, care must be taken in regard to input/output formulation, accuracy requirement, robustness and efficiency of the neural network. As mentioned before, a table-lookup model is associated with a small set of dependent variables and a restricted curve-fitting procedure. A machine-learning model, on the other hand, can handle a very large set of input parameters without needing a polynomial fitting procedure. Further, a deep-learning model allows very flexible input and output layers. Multiple middle layers in general can be used to train very complex models in response to a relatively large database with very good accuracy. Figure 1 shows a schematic of the instability wave model to be developed for transition predictions. Tensorflow 2 [9] and Keras Library [10] are used to construct the deep learning model. Details of the developed deep learning model are given in the next two subsections.

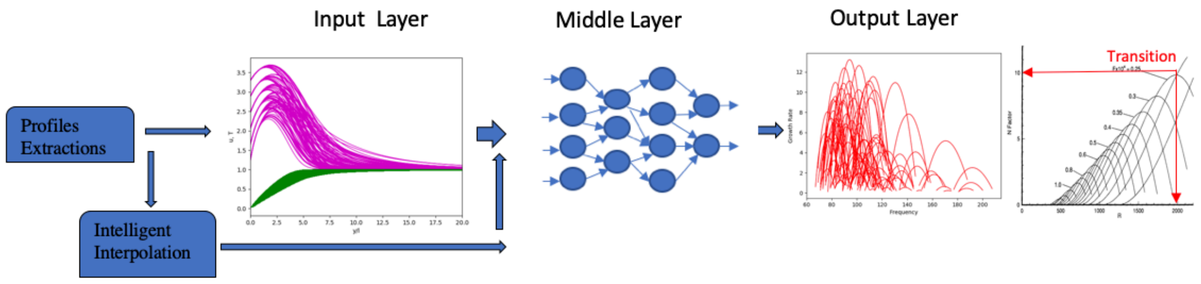


Figure 1: Schematic of the machine learning transition model.

A. Deep-learning model design and implementation

Generally, the input to the stability calculation consists of the basic flow profiles at a given location and some other global parameters such as the Mach number, freestream conditions (Reynolds number), and pressure gradient. With the instability wave calculation, one can determine the growth rates at all disturbance frequencies and wave numbers. To model this behavior, three main layers of the machine learning model are discussed as follows.

1. Input layer
   The input to the model is the normalized profiles ($[y, \rho, u, v, w, T, Re_x, \partial p/\partial x, M_e]$) at any given location of the boundary layer. The boundary-layer length scale (l) is used to normalize wall-normal distance y. Freestream conditions are used to normalize all flow variables. These profiles typically contain 100 to 200 points along the wall normal direction in the stability calculations. A fixed number of points (results shown below use 60 points in the model) is used in the designed model. For inputs with a different number of points, interpolation is first carried out across the boundary layer. In the initial phase of this research, dimensional $[y, \rho, u, v, w, T]$ profiles were used in the input layer. This choice resulted in an enormous number of training data points required because of all possible combinations of unit Reynolds numbers, streamwise and wall-normal coordinates. Using a normalized profile and Reynolds number together in the model greatly reduces the number of training data required and significantly improves the numerical accuracy. The addition of pressure gradient and Mach number is to be used by the model to tune the weights for flows under strong favorable or adverse pressure gradients such as an airfoil or in crossflow instability wave computations. The input Mach number helps the model to extend the speed range. Since no convolutional neural network is used (see the discussion below), the input layer is first flattened to a one-dimensional array as the input.
2. Middle layer
   The middle layer plays a crucial role in a number crunching machine learning model. A more complex model with a large array of output data requires more middle layers to resolve. Some numerical machine learning models construct the input layer as a two-dimensional array and use convolutional middle layers with pooling [9] to extract features for more accurate classification. Preliminary experiments in the early stage of this research found no advantages to this approach and the model was less efficient for the instability wave model. For this reason, the results shown in this paper were obtained by using four or five directly-connected middle layers by first upsampling the input array and then smoothly downsampling to the size of the output vector. For a 2D eigenvalue problem where the basic flow features are indeed two-dimensional, convolutional middle layers may be necessary. This would be a topic for future research.
3. Output layer
   For the instability wave model, there are two general choices. One can add frequency or wavenumber in the input layer and let the model just predict growth rate (e.g., Ref. [8]). This way the output layer contains just one float number. Alternatively, one can let the model predict the instability wave spectrum, or, the growth rate versus frequency vector $[f, \sigma]$. The latter was chosen due to its compactness (smaller training data size)

and its capability to quickly determine the frequency or wavenumber range for N factor integrations with a given profile. The length of this output array is set at 40 for the implemented model.

One of the most important parameters in the deep-learning model for number crunching is perhaps the activation function. An activation function takes the weighted-sum input (w) of a node from the previous layer and sends out an output to the next connected layer. The most commonly used activation function is the rectified linear unit (ReLU) function mathematically expressed as

$$A(w) = \max(0, w).$$

This function returns the input number when it is positive and zero when the it is negative. In many pattern recognition applications, this function makes certain the final value in the output layer is always a positive number. For the instability wave model, numerical experiments have indicated that a ReLU function often fails to converge the model during the training stage or results in negative frequencies in the predicted output. After trial and error, the best activation function identified is the exponential linear unit (ELU) or scaled exponential linear unit (SELU) functions defined as

$$A(w) = \lambda \begin{Bmatrix} w & for\ w \geq 0 \\ \alpha(e^w - 1) & for\ w < 0 \end{Bmatrix}.$$

Note that for ELU, the value of λ is just one. This activation function allows the output to be negative, which is desirable for the instability wave model since the predicted growth rate could be negative, meaning a stable boundary layer to the given frequency. Performances of both ELU and SELU are nearly indistinguishable for the test cases investigated in this paper. In general, the values used are $\lambda = \alpha = 1$ for the instability wave model investigated here.

Using the Tensorflow and Keras framework, the resulting model is summarized in the following code snippets.

```
model = keras.Sequential([
    keras.layers.Dense(2560, activation=tf.nn.selu),
    keras.layers.Dense(1280, activation=tf.nn.selu),
    keras.layers.Dense(640, activation=tf.nn.selu),
    keras.layers.Dense(320, activation=tf.nn.selu),
    keras.layers.Dense(164, activation=tf.nn.selu),
    keras.layers.Dense(82, activation=tf.nn.selu),
    keras.layers.Reshape((41,2))
])
optimizer = tf.keras.optimizers.RMSprop(1.e-6)
model.compile(optimizer=optimizer,loss='mse',metrics=['mae', 'mse'])
```

Note that the output layer has added one more number in addition to the output array $[f, \sigma]$. This additional number can be any physical quantity that needs to be predicted in addition to the spectrum. In the above example, five connected middle layers (excluding the last output layer) are used. With an SELU activation function for all layers. The convergence of the model is measured by the changes in root mean square (rms) of the model variations. The loss function in the model is set to be mse, which means the model training process would compute the mean squared errors between the labels and the predictions.

B. Model training and verification

Training data of the model were obtained by using either boundary-layer codes for weak inviscid-viscous interaction boundary layers or a Navier-Stokes solver otherwise. Since the model is profile based, the required training data size is smaller compared to models trained by gross boundary-layer parameters and varying disturbance frequencies and/or wave numbers. The inclusion of pressure gradient is needed for a future parabolized stability equations (PSE) model and can be used to improve the overall accuracy for quasiparallel linear stability theory (LST) models as well. As mentioned above, multiple fully-connected middle layers are needed to improve the numerical accuracy for various instability wave mechanisms such as Tollmien-Schlichting (TS), Mack's mode, or crossflow instability.

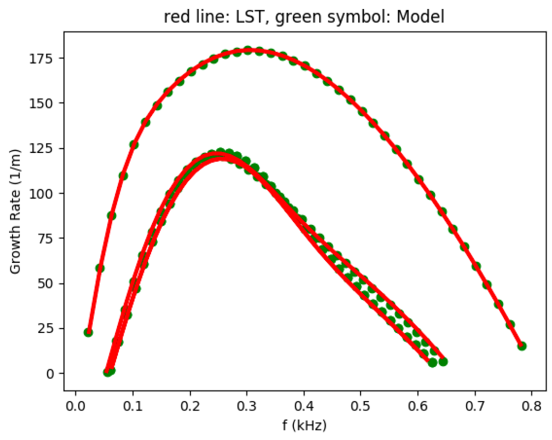
A flow instability database is also needed to train the deep learning model. In this study, meanflow solutions for a range of Mach numbers with boundary-layer edge conditions pertinent to TS, crossflow, first- and second-mode instabilities are first computed using a compressible boundary-layer code [11] for flat plates, airfoils, and sharp-cone configurations with a Mach number varying from 0.01 to 6. Meanflows for infinite swept wing boundary layers with crossflow instability are generated by the BLSTA code [12]. A blunt cone with a nose radius of 2.5 mm, a length of 1.1 m, and a Mach number ranging from 4 to 6 is also computed with the Navier-Stokes flow solver VULCAN-CFD [13]. Profiles from the meanflow database are extracted and fed into the deep learning model described above. For model training, the labels (instability spectrum as an array of $[f, \sigma]$) for each set of flow profiles is precomputed by using LASTRAC [4], a well-establish physics-based transition prediction software. Flow conditions for all the training cases are listed in Table 1. This selection is by no means comprehensive. They were chosen to demonstrate the effectiveness of the model in handling most common types of instability waves across the speed regime in some practical applications. For the ease of data generation, the Python library front end for LASTRAC (PyLASTRAC) to be described in more detail later was used to analyze stability of all meanflow cases and write out the computed instability wave spectrum.

Table 1. Flow conditions for all test cases used for model training and verification

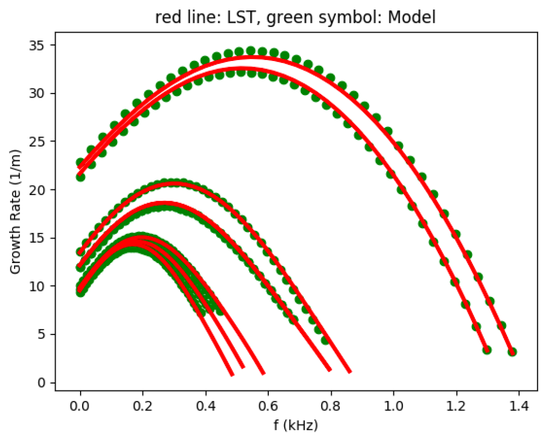
Case	**Instability type**	**Mach number**	**Thermal condition**
Blunt cone	Second mode	4-6	$T_w = 300 - 800K$
Sharp cone	Second mode	6	$T_w/T_{adw} = 0.2 - 1.0$
Flat plate	First mode	2	$T_w/T_{adw} = 0.4 - 1.6$
Flat plate	Second mode	4-7	$T_w/T_{adw} = 0.2 - 1.0$
Flat plate	TS wave	0.01-0.9	$T_w/T_{adw} = 1.0$
NLF 2D airfoil	TS wave	0.01	$T_w/T_{adw} = 1.0$
NLF infinite swept wing	Crossflow instability	0.01	$T_w/T_{adw} = 1.0$

For the model training, 70% randomly selected profiles among all meanflow profiles generated are chosen as the training set. For each input in this selected set, the known instability wave spectrum is called a label. Model training is carried out in an iterative manor by adjusting weights of all neurons such that the norm of errors between the computed output array and the labels are minimized. The rate of convergence strongly depends on the complexity of the input training data, typically, it takes about 6,000-10,000 iterations to reach an L2 norm of about 1.e-5 or smaller. Convergence rate for a dataset with a single instability mechanism is in general much faster than the mixed type. The remaining 30% of the dataset is used to evaluate the accuracy of the model. Normally, we can feed a few randomly selected meanflow profiles from this dataset into the model to make certain the predictions are indeed very close to the corresponding labels from the dataset. When the L2 norm of error drops to 1.e-5 or smaller, the accuracy is normally quite good. Figure 2 shows some representative results for various instability mechanisms with different flow speeds. For each selected case, a meanflow profile is generated. Using the trained-model, the predicted instability wave spectrum is compared with the LASTRAC results in the figure. The results show good overall agreements of model predicted and linear stability theory (LST) computed instability wave spectra for low-speed TS wave, crossflow instability and second-mode disturbances for a Mach 5 blunt cone.

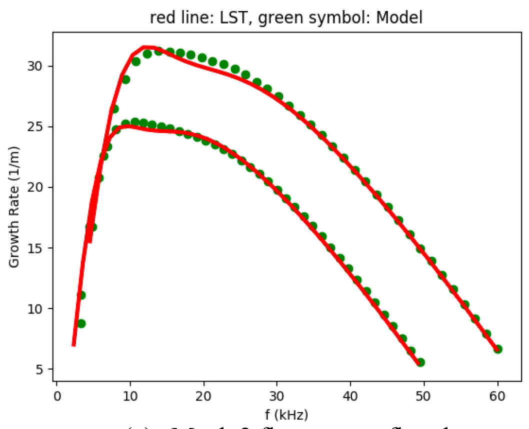
Results shown above demonstrate that it is quite feasible to employ and train a deep-learning model to reproduce the physics-based predictions based on linear stability theory with a good overall accuracy. The capability to predict an unstable frequency range for any given profiles offers a useful tool for transition prediction. It could be used to replace any table lookup procedure with better accuracy. If the labeling of the training data set includes the stable region, the trained model could potentially replace all LST codes for N factor integration along any trajectories in a 3D boundary layer. Model training using linear PSE data has not been attempted in this research. With the increased fidelity of the stability data from PSE computations, more data points are needed for the training to effectively incorporate nonparallel effects and other geometrical variations such as curvature. This topic will be left for future research. The current developed model is intended to be a proof-of-concept only. As the machine learning model is much more sensitive to number crunching than classification, it is expected that to generate a universally comprehensive deep-learning model for instability wave predictions, the training data set must be as inclusive as possible so that a very broad range of Mach number, Reynolds number, thermal conditions, and pressure gradients is covered for practical applications.



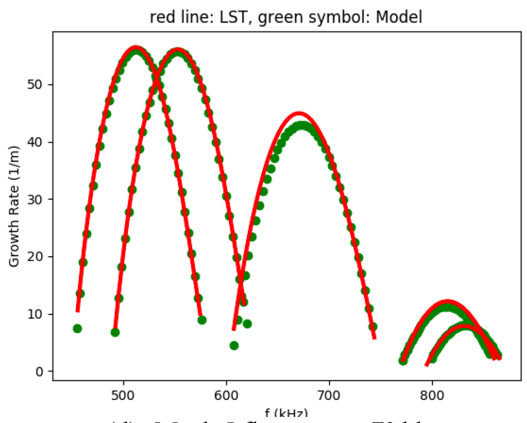
(a) 2D airfoil



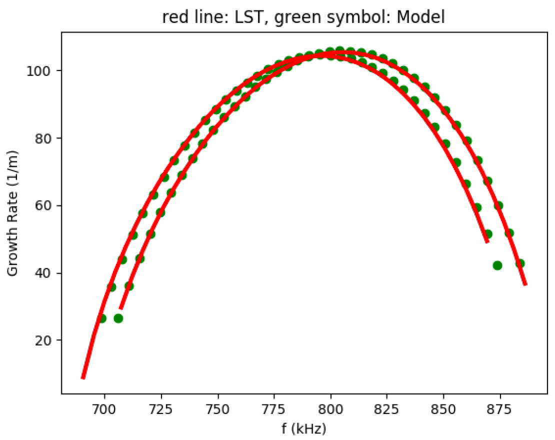
(b) 45° swept wing



(c) Mach 2 flow over a flat plate



(d) Mach 5 flow over a 7° blunt cone



(e) Mach 6 flow over a 7° sharp cone

Figure 2: Comparison of model prediction (symbols) and theory (lines) for several selected input profiles at several representative streamwise locations where the mean flow is unstable: (a) TS wave, (b) crossflow, (c) first mode (d) second mode (blunt cone), and (e) second mode (sharp cone).

III. Intelligent Interpolation Model

When applying the trained instability model to Navier-Stokes codes for transition prediction, it is very likely that the boundary layer is often underresolved, especially in low Reynolds number (Re_x) regions where the boundary layer is relatively thin. A similar deep learning model was also devised to perform intelligent interpolation of boundary-layer profiles for a very small number of points across the boundary layer. Interpolation of boundary layer profiles are usually done by using a spline curve fit. Higher order splines are able to fit the profiles more accurately. However, if the input is too sparse, overfitting often occurs and produces the wrong instability-wave predictions due to erroneously produced inflectional points. A more intelligent machine learning model, in contrast, can interpolate the profiles that belong to the same family of the developing viscous boundary layer, provided proper training is done before hand. The input to this model is similar to that described above in the instability waves model, but with only around 5-10 discrete points in the profiles. The model is typically implemented as follows.

```
def make_model():
        model = tf.keras.Sequential()
        model.add(tf.keras.layers.Flatten())
        model.add(tf.keras.layers.Dense(1010, use_bias=False))
        model.add(tf.keras.layers.LeakyReLU(alpha=0.2))
        model.add(tf.keras.layers.Dense(505, use_bias=False))
        model.add(tf.keras.layers.LeakyReLU(alpha=0.2))
        model.add(tf.keras.layers.Reshape((101, 5)))
        return model
model = make_model()
optimizer = tf.keras.optimizers.RMSprop(1.e-6)
model.compile(optimizer=optimizer,loss='mse',metrics=['mae', 'mse'])
```

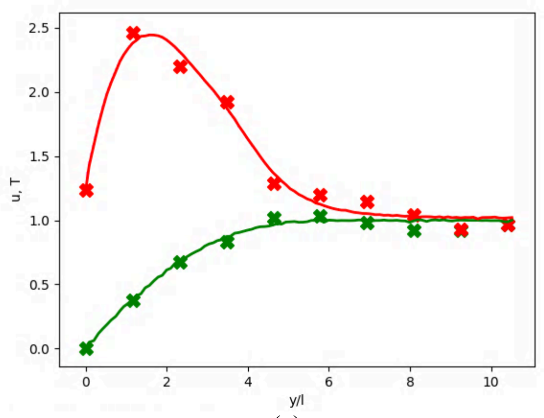
Here, the output layer consists of an array, $[y, \rho, u, v, T]$, with 101 points for each variable. The leaky ReLU activation function,

$$A(w) = \begin{Bmatrix} w & for\ w \geq 0 \\ \alpha w & for\ w < 0 \end{Bmatrix}$$

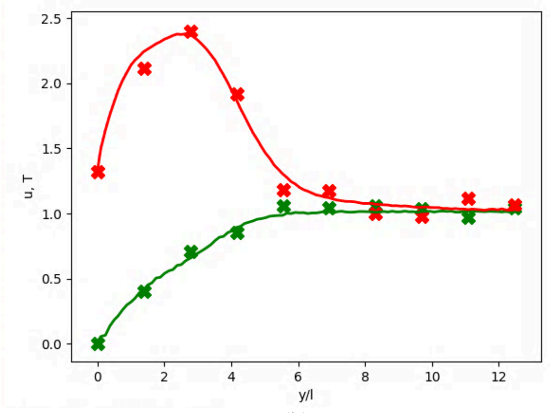
with $\alpha = 0.2$ was found to work the best.

As a demonstration of the effectiveness of this model, meanflow profiles for the blunt cones are used to train the model. For each profile in the training data, the input layer consists of a 10-point profile subsampled from the full profile (around 200 points) and the full profile serves as the label for training purposes. After the model has been trained, several randomly selected full profiles are selected as the testing databases. Each profile in the testing databases is subsampled to 10 points with velocity and temperature values randomly shifted by a small amount for all points in the profile. This shifting is to mimic inaccuracies of the Naviser-Stokes solver due to insufficient resolution. Testing data are then fed into the trained intelligent interpolation model for predictions (interpolated, fine profiles with 101 points). The interpolated profiles subsequently are fed into the instability wave model to predict the instability wave spectrum. Figure 3 shows representative results for the input 10-point shifted profile and a comparison of the instability wave prediction with that obtained from the original highly resolved profile calculated using LASTRAC. As can be seen, for a mere 10 points in the inaccurate profiles, the intelligent interpolation model was found to be able to interpolate to a 101-point more refined profile with good accuracy such that the predicted instability wave spectra agree reasonably well with that predicted by a full profile and LST computations.

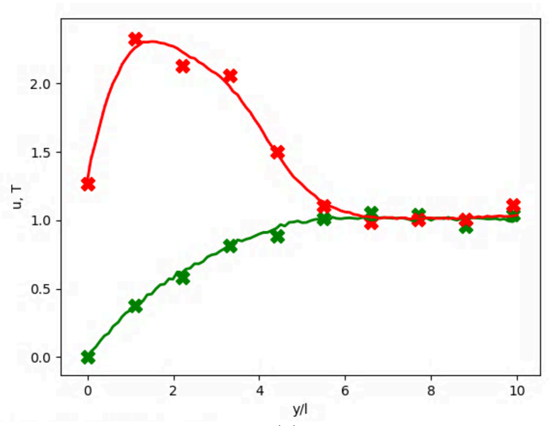
These results are quite encouraging because conventional wisdom indicates that a set of underresolved profiles like those shown in Fig. 3 were useless for instability wave predictions. And yet, through an intelligent interpolation model, the instability wave spectra computed for these profiles are quite reasonable for practical use in transition predictions. The above interpolation model has also been found to work well to interpolate a turbulent (RANS) profile so that the model predicted eddy viscosity agrees well with predictions from the full-fledged RANS model with only 11 discrete points in the turbulent meanflow profiles. This topic is beyond the scope of this paper and is left for a future study.



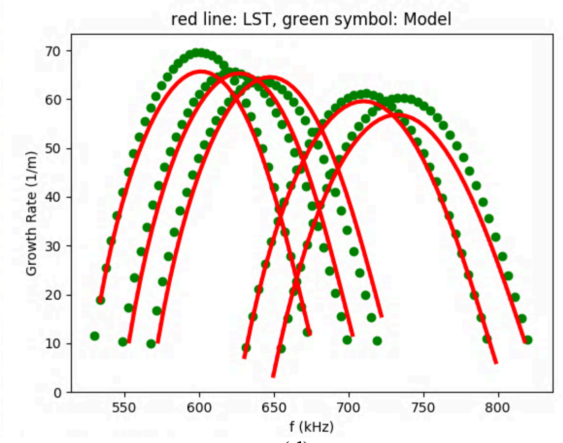
(a)



(b)



(c)



(d)

Figure 3: Intelligent profile interpolations for selected hypersonic blunt cone meanflow profiles, symbols are the shifted 10-point profiles and lines are the model interpolated boundary layer profiles with temperature in red and streamwise velocity in green: (a)-(c) three different profile interpolations, (d) comparison of model-predicted instability spectra using the interpolated and exact profiles predicted by LASTRAC (symbols are predictions and lines are exact solutions).

IV. Interfaces with Existing CFD Codes

A well-designed software interface library is required for the communication between the developed model and the CFD solver. Similar interface implementation between a RANS code and an LST solver has been developed for the unstructured-mesh TAU CFD code with focus on industrial applications in the literature (e.g., Refs. [14-15]). One can implement the interface routines inside the CFD solver by directly utilizing its data structures. Doing so would not incur much more resource overhead and could impose the predicted transition fronts more directly. The major disadvantage is the extra efforts needed when a profile extraction goes from processor to processor in a parallel computing framework. As discussed in the Introduction Section, a transition model based on table lookup often uses the existing CFD solver infrastructure to integrate N factor by solving a transport equation on the same grid as the CFD solution. For physics-based models, this approach would not work due to several reasons. First, formulation and solution of stability equations requires a wall-normal profile and a specific spanwise direction commensurate with the instability mechanism at any given viscous surface point, which does not exist in the CFD mesh most of the time. Second, N factor integration as the basis of transition onset determination only makes sense along a physical trajectory such as a streamline, a vortex or group-velocity line. Points along these lines would not coincide with the CFD mesh. Third, physics-based transition prediction only works in the laminar region. Once a transition front is determined in the RANS transition model, the CFD solution in the turbulent region renders useless to the instability calculations. The laminar extent of the instantaneous solution changes according to the physics-based model predictions. If the transition fronts are set prematurely to be too close to the leading edges during the CFD solver iteration, the integrated

N factor distribution may not give rise to transition at all in the following iterations due to a too-small laminar region. Consequently, unlike the RANS-based or table lookup models, the physics-based transition model is only to be engaged infrequently, there is no need for a tight integration with the CFD code infrastructure. A very loosely coupled approach requires the passing of only the instantaneous CFD solution on the entire 3D volume mesh, and in return, the physics-based model passes the transition markings on the 3D surface back to the CFD solver. As such, a set of interface tools has been developed to allow efficient communications between the CFD solver and the LST/PSE based transition prediction tools. Figure 4 depicts how this model works. In the case of multidomain parallel CFD computations, the volume-mesh instantaneous solution needs to be assembled before passing to the model. Similarly, the transition marking on all the wall vertices needs to know its way back to different processors. These additional efforts are minimal and only need to be done once for each CFD computation. In the following subsections, key components are discussed in more detail. For illustration purpose only, the incompressible flow over a prolate sphere [14-15] with an angle of attack has been chosen for investigation using the developed interface. Laminar or RANS computations of this configuration have been carried out by the FUN3D [16] and OVERFLOW [17] solvers.

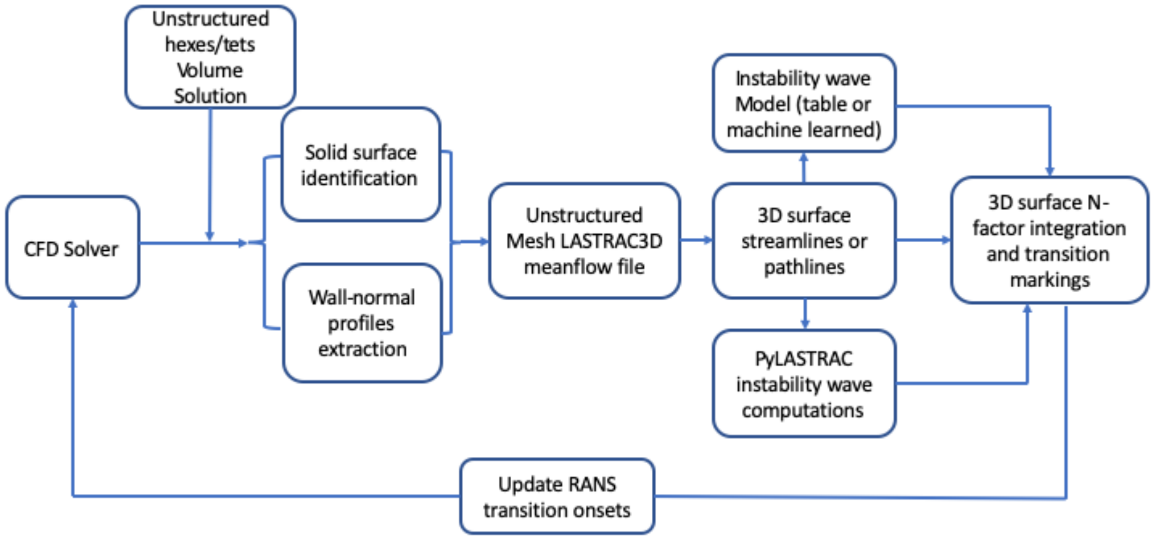

Unstructured hexes/tets Volume Solution
CFD Solver
Solid surface identification
Wall-normal profiles extraction
Unstructured Mesh LASTRAC3D meanflow file
Instability wave Model (table or machine learned)
3D surface streamlines or pathlines
PyLASTRAC instability wave computations
3D surface N-factor integration and transition markings
Update RANS transition onsets

Figure 4: Schematic of the interface of the physics-based transition model to CFD solver.

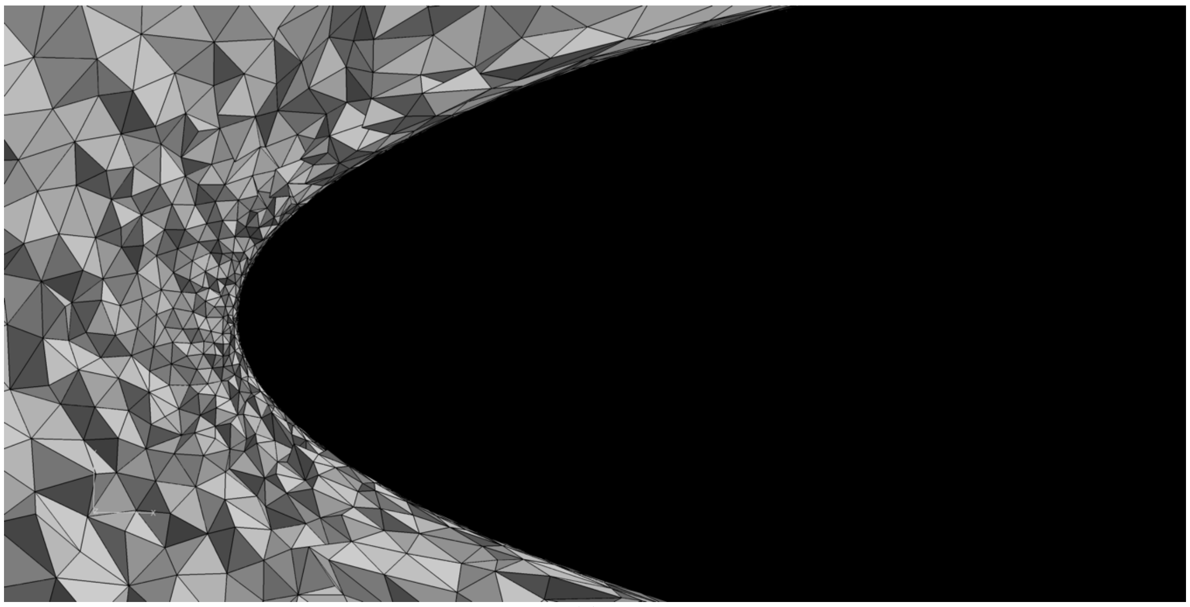
A. Profile extraction tools

Currently, an unstructured-mesh volume solution format is chosen as the input to the interface tool due to the rising popularity of unstructured-mesh CFD solvers. This choice makes it easier to pass CFD solutions to the interface for unstructured-mesh solvers. For structured-mesh CFD solvers, it is straightforward to combine multi-block structured-mesh grid and solutions into a single-block unstructured hexahedral mesh. The developed interface tool set includes a code to convert multiblock PLOT3D [18] grid and solution files. In the case of overset grids, users must determine which grid/solution set has a higher priority for profile extractions. In cases with a very large number of parallel blocks and combining blocks is very time consuming, additional codes need to be developed to automatically navigate through different files to access volume solutions generated by each block.

From the given volume solution, all solid surface vertex nodes are first extracted and reorganized to form a 3D unstructured triangular or quadrilateral surface mesh. The next step is to extract wall-normal profiles at each surface vertex point. Two different algorithms are used, discrete points or exact normal. In both approaches, the exact normal $\vec{n}$ at any given surface vertex is first determined by using the known surface geometry from the 3D surface definition already extracted. For a given vertex point P_i along the line of extraction, the next point P_j is determined by the following algorithm:

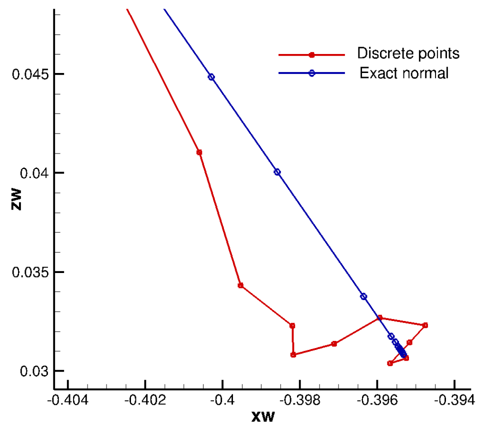
$$D(p,\vec{n}) \coloneqq \ distance\ between\ point\ p\ and\ \vec{n}$$
$$P_{ik} \coloneqq \{P_k \mid vertex\ P_k\ is\ an\ m-layer\ neighbor\ of\ P_i\}$$
$$j = \arg\min_k [D(P_{ik},\vec{n})]$$

where an m-layer neighbor is defined by searching immediate neighbors m times outward around the point P_i. The set of discrete points for profile extractions starts from a given surface vertex and the search stops when it reaches the domain boundaries. Flow solutions at these discrete points are used for the profiles after the velocity vector is projected onto the exact normal direction. Normally, for a well-organized and nearly wall-normal grid line extruded from the surface, the above approach would yield quite accurate wall-normal profiles. The second approach is to take all the discrete points obtained from the first approach and project all of them onto the exact normal line. These projected points all fall exactly on the wall-normal direction. Profiles are then extracted for these projected points using volume element interpolation, i.e., locate every point within a volume element and perform in-element interpolation to ensure accuracy. The exact-normal approach is more accurate than the discrete points approach in general. However, care must be taken in accurately locating the volume element for each point and also in making sure the normal line does not go out of bounds near the domain boundaries. Figure 5 shows an example of a coarse and stretched tetrahedral mesh near the surface along with the points used for extraction and the extracted profiles from both approaches for comparison. The stretched obtuse tetrahedral elements near the surface result in zig-zag discrete points. On the other hand, the exact normal approach gives a very smooth distribution of extraction points. The interpolated velocity profile from the zig-zag points is in fact not too far off the exact-normal results after they are projected onto the normal. This is not surprising due to the fact that the flow usually changes much more slowly along the streamwise than along the wall-normal direction for an attached boundary layer. For flows with separation, this may not be the case and the exact-normal approach would offer a more robust profile extraction in general. Note that for the same configuration, a structured hexahedral mesh with well-resolved and nearly normal gridlines, the discrete-points and exact-normal approaches yield nearly identical extracted profiles.

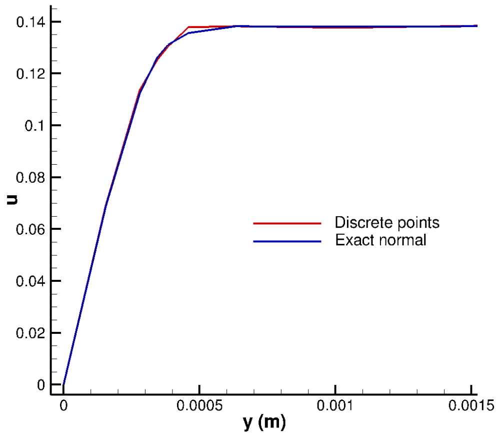


(a)

Figure 5: Example of elongated obtuse tetrahedral CFD mesh near wall: (a) near wall mesh, (b) field points in Cartesian coordinates (xw and zw) used to extract profiles by using discrete points or exact normal, and (c) comparison of extracted normalized velocity profiles.



(b)

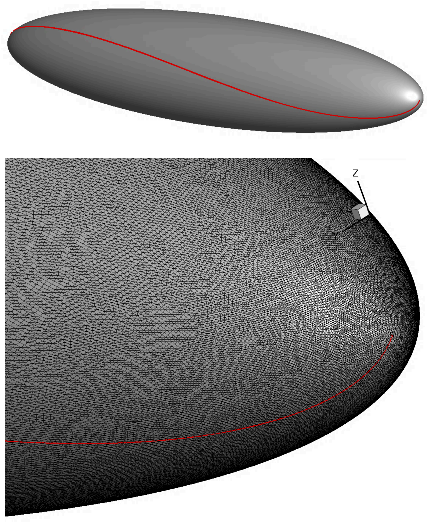


(c)

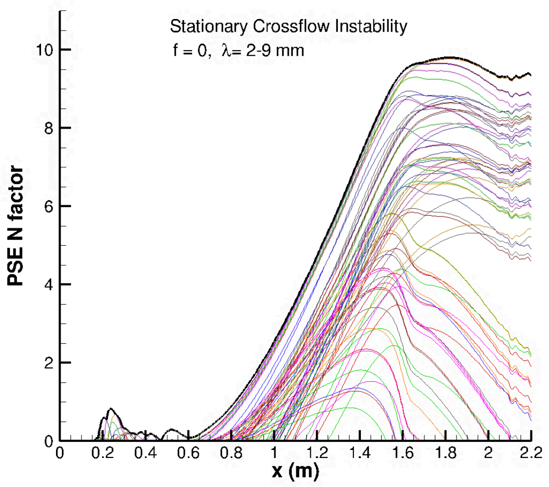
Figure 5: Concluded.

B. Unstructured-mesh LASTRAC Implementation

To handle the meanflow profiles extracted in the previous section in the unstructured mesh format, new capabilities have been implemented in the LASTRAC software. The original format for 3D boundary layers is limited to meanflows that can be expressed as a collection of wall-normal profiles at each vertex of a structured $M \times N$ quadrilateral surface mesh, where M and N are the grid dimensions in the streamwise and spanwise direction, respectively. The newly developed LASTRAC 3.0 version to be released soon added an option to input an unstructured triangular or quadrilateral surface mesh with wall-normal profiles defined at each surface vertex. The new format applies to both perfect-gas or nonequilibrium chemically reacting flows. All LST, linear and nonlinear PSE methods except the surface marching PSE remain available for unstructured-mesh meanflows along various user-selected 3D integration paths including streamlines, vortex lines, group velocity lines, and user-defined pathlines. Both triangular and quadrilateral (but not mixed) unstructured surface meshes are allowed. Meanflow velocity components for the unstructured format are now in Cartesian coordinates instead of the transformed contravariant components required in the original structured-mesh format. A tool to convert the old structured-mesh format into the unstructured-mesh format is also available. Figure 6 demonstrates the PSE N-factor results along a streamline from a typical run with a triangular surface mesh. A streamline is initiated in the middle section of the prolate sphere body. The complete streamline starting from the stagnation point is generated via both upstream and downstream tracing. Linear PSE computations are then carried out after identifying the unstable spanwise wave number range. The envelopes of N values versus the streamwise distance along the streamline are formed for transition front prediction to be discussed in the next subsections.



(a)



(b)

Figure 6: Incompressible flow over a prolate sphere with an angle of attack, run with LASTRAC unstructured triangular surface mesh: (a) Streamline and surface mesh near the leading edge, and (b) Linear PSE envelopes (dark line) for stationary crossflow instability.

C. N-factor Integration over arbitrary 3D surfaces

Integration of disturbance growth in terms of N factor as shown in Fig. 6(b) is an approximation because only the disturbance evolution along the streamwise direction is fully accounted for and the effects of basic flow variation along the spanwise direction is ignored. In other words, as in Eq. (1), disturbance variation in the spanwise direction is fully described by $e^{i\beta z}$ with a real β, implying locally spanwise periodic. Nonetheless, this local line marching approach is a good approximation as compared to solutions obtained from higher fidelity methods in 3D boundary layers [19]. In several applications across the speed regime, N-factor distribution via a local line marching PSE method along the streamlines agrees reasonably well with experimentally measured transition fronts (e.g., Refs. [19-22]). Hence, the transition prediction module of the developed interface tool set uses LST or PSE computations along streamlines to map out the transition onsets over a 3D boundary layer, though other options such as along the vortex or group velocity line for crossflow instability is also available. The selection of streamlines should be such that they give good coverage over the entire body. Figure 7 shows two examples of streamlines generated by employing only 60 randomly selected vertex seeds. The entire streamline is traced from either the stagnation point or the attachment line all the way to the trailing edge for each vertex seed.

Disturbance evolution along the generated streamlines needs to be determined for all unstable ranges of disturbance frequencies or wave numbers before the surface N-factor distribution can be determined. Depending on the freestream conditions, all instability mechanisms must be accounted for. For low speed flows over a 3D body with an angle of attack, both TS wave and crossflow instability can lead to transition. Along a given streamline, the instability range can either be determined by first calculating the gross boundary-layer parameter and looking up a precalculated table or by feeding meanflow profiles in the trained machine learning model as described in Section II after the model has been well-trained for the given freestream conditions. In addition, the most accurate way to identify instability wave range and calculate N-factor envelopes is to directly invoke a transition prediction tool like LASTRAC. The wall-normal profile extraction tool described in Section IV.A can write out the unstructured-mesh LASTRAC meanflow to be used for both streamline generation and instability wave calculations over the given 3D boundary layer. Using LASTRAC in lieu of either the table lookup or machine learning model is the most robust way to perform physics-based transition prediction, at least before a comprehensive machine learning model across all instability mechanisms and speed regimes are ready to be deployed. Computational time required for direct LASTRAC runs is certainly more than that of a working machine learning model. This issue will be further discussed in the next subsection.

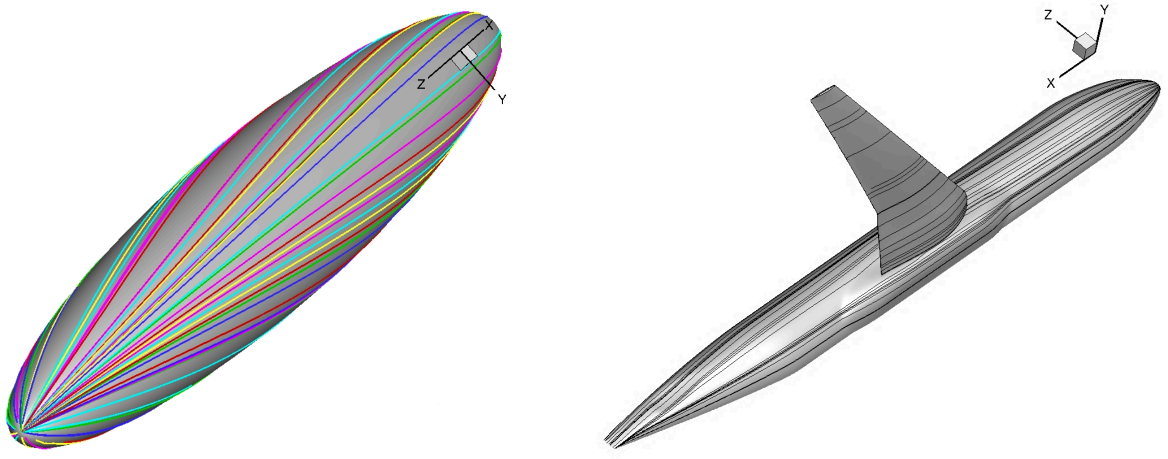
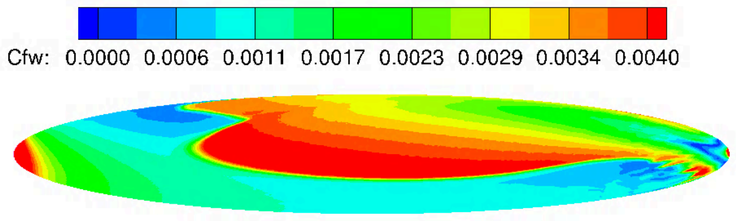


Figure 7: Streamlines generated by 60 randomly selected vertex seeds: (a) spheroid at angle of attack, and (b) fuselage and wing body.

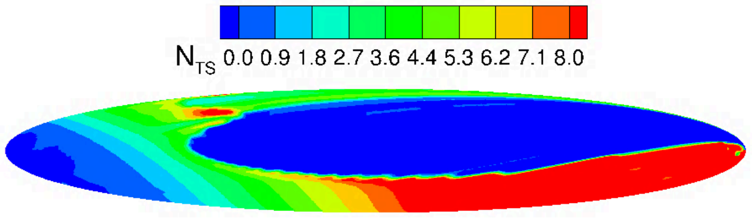
In addition to performance, user-friendliness of transition analysis for a given configuration is also of major concern. The common perception of carrying out transition prediction computations is that it requires a high level of expertise. Users need to be well versed in flow physics as well as numerical analysis in order to effectively perform required computations using transition prediction tools such as LASTRAC. To circumvent this issue and make transition prediction much more user-friendly, a set of Python [23] codes were developed as the front end to run LASTRAC for transition analysis. The tool set with the code name PyLASTRAC wraps different combinations of LASTRAC options into simple object instantiations and function calls. It can scan unstable frequency/wave number ranges for any given meanflow automatically and display results on the screen for inspections. Users can also inspect the 3D surface and the meanflow profiles written in the LASTRAC input meanflow file or the computed eigenfunction distributions if an unstable mode is found. PyLASTRAC routines encompass nearly all LASTRAC-provided options for a 2D/axisymmetric, 3D structured, or unstructured-mesh boundary layer. For any given surface vertex point, PyLASTRAC can be used to carry out streamline formation, instability wave identification, LST or PSE runs, and N-factor envelope calculations. At the highest level, PyLASTRAC can be executed from the command line to automatically complete all the tasks for user-provided meanflow file and seeded surface vertex indices. With this option, the N-factor distribution over a 3D surface can be carried out with almost no user intervention. Currently, it has been tested mainly for subsonic flows over a 2D/axisymmetric body, a wing, or slender body. Continued development and testing to improve the user-friendliness would expand the applicable regimes in the near future.

D. Interface software integration

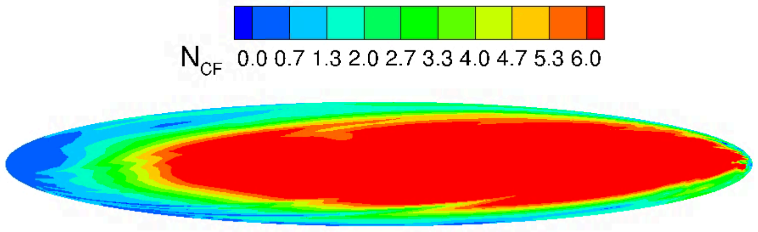
All components described above have been integrated into a single workflow Python script. Only information such as volume mesh type, normalization parameters, body type (slender or wing), desired output file names, and computing resources is needed for the workflow script to execute all steps depicted in Fig. 4. If desired, the workflow script can be embedded into a callable C++ program, which is generally accessible from CFD solvers. However, as discussed above, if only sparing communications with the CFD solver are needed, detached runs with the integrated Python script after the instantaneous laminar or RANS solution has been written should work just fine. In the following example runs, an incompressible transitional flow over a 6:1 prolate sphere at 10° angle of attack previously investigated both experimentally and computationally [24] was chosen to test the developed interface suite of codes. The OVERFLOW [17] CFD solver with the SST-2003-LM2015 transition model [25] was used to compute the meanflow with a structured mesh. The converged transitional flow solutions were fed into the developed physics-based transition model for transition prediction. The integrated workflow script was able to execute all steps fully automatically. The PyLASTRAC software ran LASTRAC code autonomously to predict both TS wave and crossflow instability for 40 streamlines. Figure 8 shows the predicted crossflow and TS wave N-factor distribution over the spheroid, along with the friction coefficient distribution predicted by the SST-2003-LM2015 transition model. As shown, a mixed-mode transition is evident with TS waves dominating only near the upper and lower symmetry planes, while stationary crossflow instability may lead to transition around the middle. The current model predicts earlier (more upstream) transition than that from the OVERFLOW RANS computations with the transition model. Transition fronts predicted using the picked N values shown in Fig. 9 agree reasonably well with the experiments. The dip of current predicted transition fronts from 140° to 170° is associated with the transitional state around that region from the RANS meanflow. It is expected to improve with iterative solutions. A better way to predict mixed-mode transition would require a more sophisticated $N_{CF} - N_{TS}$ model, which is left for future investigations along with more iterations between the CFD solver and the physics-based transition model predictions.



(a)



(b)



(c)

Figure 8: Comparison of friction coefficients from RANS and current linear PSE predicted crossflow and TS wave N-factor distributions over the 6:1 prolate sphere: (a) friction coefficient from RANS transition model, (b) N-factor distribution for TS waves, and (c) N-factor distribution for stationary crossflow instability.

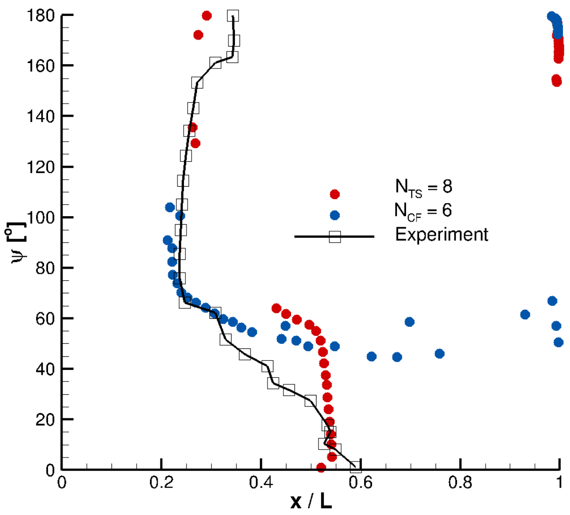


Figure 9: Comparison of predicted transition fronts on the 6:1 prolate sphere using linear PSE predicted $N_{CF} = 6$ and $N_{TS} = 8$ with experimental measurements [24].

Note that the most time-consuming modules of the developed interface suite are the profile extraction and the LASTRAC computations. Fortunately, both computations are perfectly parallelizable. Profiles extraction at each surface vertex point is totally independent. The code developed takes advantages of both multi-CPU and multithread. Within each processor, multithread computations can further speed up the computations. The index searching process for the interpolation of wall-normal points can be recorded for repeated usage during the iterative communications with the CFD solver. The LASTRAC software has multithread options built in for instability wave computations and N-factor integration along a given streamline. Multiple streamlines can be run in a parallel cluster environment to significantly improve the overall performance since all streamlines are independent of each other. These parallel computations can be executed using the Python script without the need to invoke any message passing interface (MPI) routines. With an efficient fully parallelized transition computation using LASTRAC, the machine learning model discussed previously may or may not help significantly improve the overall efforts required for physics-based transition prediction, especially when a tremendous amount of work is needed to train the model in order to ensure a good coverage for all practical applications. More numerical experiments are needed in the future to evaluate the pros and cons of using a machine learning instability wave model.

V. Concluding Remarks

Substantial progress has been made in the development of a set of computational tools that provide on-the-fly physics-based transition prediction for a given laminar or transitional flow from a RANS solver with either structured or unstructured meshes. The loosely coupled communication between the CFD solver and the transition computation modules is through passing very generic unstructured-mesh volume solution data. The interface tool set is invoked to convert the passed data into an unstructured-mesh LASTRAC meanflow file, with which an efficient computation of instability waves and N-factor integration can be carried out under a parallel cluster environment. Based on the computed N-factor distribution, all 3D surface vertex points can be marked with either a laminar or turbulent state. Such transition markings can then be passed back to the CFD solver for the update of transition fronts over the body and the iteration continues until full convergence. In general, the developed physics-based transition model provides a viable alternative to RANS-based transition models for existing CFD solvers. Although more work is needed, the former is likely to deal with mixed-mode transition cases better due to its compliance to transition physics.

A machine learning instability wave prediction model in conjunction with an intelligent interpolation deep learning model has also been investigated. It was found that with multiple middle layers in the neural network, it is possible to devise a machine learned model that can predict multiple instability wave mechanisms with a Mach number ranging from 0 to 7 and to interpolate an underresolved meanflow profile to produce reasonable instability wave prediction. Deep learning algorithms identified can be used for the construction of a comprehensive instability wave model that covers all flow conditions in practical applications in the future.

Acknowledgments

The funding for this research has been provided by the Revolutionary Computational Aerosciences (RCA) subproject under the Transformational Tools and Technologies (TTT) project of the NASA Transformative Aeronautics Concepts Program. The author would like to acknowledge Dr. Fei Li and Dr. Nate Hildebrand of the NASA Langley Research Center and Dr. Balaji Venkatachari of National Institute of Aerospace for providing the blunt cone meanflow database for model training and the unstructured- and structured-mesh prolate sphere meanflows.

References

[1] Langtry, R. B. and Menter, F. R., "Correlation-Based Transition Modeling for Unstructured Parallelized Computational Fluid Dynamics Codes," *AIAA J.*, Vol.47, No. 12, 2012.

[2] Langtry, R. B. and Menter, F. R., "Transition Modeling for General CFD Applications in Aeronautics," AIAA Paper 2005-522, 2005.

[3] Coder, J. G. and Maughmer, M. D., "A CFD-Compatible Transition Model Using an Amplification Factor Transport Equation," AIAA Paper 2013-0253, 2013.

[4] Chang, C.-L., "Langley Stability and Transition Analysis Code (LASTRAC) Version 1.2 User Manual," *NASA TM2004-213233*. URL https://ntrs.nasa.gov/search.jsp?R=20040082550.

[5] Slotnick, J., Khodadoust, A., Alonso, J., Darmofal, D., Gropp, W., Lurie, E., and Mavriplis, D., "CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences," NASA Technical report, NASA/CR-2014-218178, 2014.

[6] Sturdza., P., “An Aerodynamic Design Method for Natural Laminar Flow Aircraft,” Ph.D. Dissertation, Stanford University, Dec. 2003.

[7] Davis, M. B., Reed, H. L., Youngren, H., Smith, B., and Bender, E., “Transition Prediction Method Review Summary for the Rapid Assessment Tool for Transition Prediction (RATTraP),” Technical Report AFRL-VA-WP-TR-2005-3130, Air Force Research Lab, Wright-Patterson AFB, OH, 2005.

[8] Paredes Gonzalez, P., Venkatachari, B. S., Choudhari, M., M., Li, F., Chang, C.-L., Muhammad I. Irfan, and Xiao, H., “Towards a Practical Method for Hypersonic Transition Predictions based on Stability Correlations,” *AIAA J.* 3 Aug 2020 https://doi.org/10.2514/1.J059407.

[9] “Tensorflow,” https://www.tensorflow.org/, 2021 (last accessed 6/3/2021).

[10] “Keras,” https://keras.io/, 2021 (last accessed 6/3/2021).

[11] Harris, J. E. and Blanchard, D. K., “Computer Program for Solving Laminar, Transitional, or Turbulent Compressible Boundary-Layer Equations for Two-Dimensional and Axisymetric Flow,” NASA Technical Memorandum 83207, 1982.

[12] Wie, Y.-S., “BLSTA—A Boundary Layer Code for Stability Analysis,” NASA CR 4481, 1992.

[13] “VULCAN,” http://vulcan- cfd.larc.nasa.gov/, 2016 (last accessed 02/07/2018).

[14] Krimmelbein, N., and Radespiel, R., “Transition Prediction for Three-Dimensional Flows Using Parallel Computation,” *Computers & Fluids*, Vol. 38, No. 1, 2009, pp. 121–136.

[15] Krimmelbein, N., and Krumbein, A., “Automatica Transition Prediction for Three-Dimensional Configurations with Focus on Industrial Applications,” *J. Aircraft*, Vol. 48, No. 6, pp. 1878-1887, 2011, doi: 10.2514/1.C031230.

[16] Anderson, W. K., and Bonhaus, D. L., “Implicit Upwind Algorithm for Computing Turbulent Flows on Unstructured Grids,” *Computer and Fluids*, Vol. 23, No. 1, 1994, pp. 1–22.

[17] Nichols, R. H, and Buning, P. G., “User’s Manual for OVERFLOW 2.3, Version 2.3,” NASA Langley Research Center, Hampton, VA, Oct 2019.

[18] “Plot3d File Format for Grid and Solution Files,” https://www.grc.nasa.gov/www/wind/valid/plot3d.html, 2021 (last accessed 05/17/2021).

[19] Chang, C.-L., “LASTRAC.3d: Transition Prediction in 3D Boundary Layers,” AIAA Paper 2004-2542, 2004.

[20] Chang, C.-L., Choudhari, M., Hollis, B., and Li, F., “Transition Analysis for the Mars Science Laboratory Vehicles,” AIAA Paper 2009-4076, doi:10.2514/6.2009-4076.

[21] Choudhari, M., Chang, C.-L., Jentink, T., Berger, K., Candler, G., and Kimmel, R. L., “Transition Analysis for the HIFiRE-5 Vehicle,” AIAA Paper 2009-4056, 2009.

[22] Tufts, M. W, Gosse, R. C., and Kimmel, R. L., “Parabolized Stability Equations Analysis of Crossflow Instability on HiFiRE-5v Flight Test,” *J. Spacecraft & Rockets*, Vol. 55, No. 6, pp. 1369-1378, 2018.

[23] “Python,” https://www.python.org/, 2021 (last accessed 05/17/2021).

[24] Krimmelbein, N. Krumbein, A., and Grabe, C., “Validation of Transition Modeling Techniques for a Simplified Fuselage Configuration,” AIAA Paper 2018-0030, 2018, doi:10.2514/6.2018-0030.

[25] Langtry, R. B., Sengupta, K., Yeh, D. T., and Dorgen, A., J., “Extending the $\gamma - Re_\theta$ Correlations based Transition Model for Crossflow Effects,” AIAA Paper 2015-2474, 2015.