Notes on PSE and Adjoint PSE Code Development and Validation

S. Scott Collis

May 9, 2022

1 Validation of PSE

1. I took the Blasius profile at $R = 580$ with $\alpha = 0.179$ and computed the temporal growth-rate. The result is $\omega = 6.517833 \times 10^{-2} + \imath 1.425171 \times 10^{-3}$ which is in very good agreement with Grosh & Orszag (1997).

2. However, in comparing linear PSE to Bertolotti, I have found that my nonparallel growth-rates are too low. This must mean that I have messed up the vertical component of velocity. Note that when using the same base flow, the LPSE and HLNS were in excellent agreement which indicates that the base-flow is the culprit.

3. Currently the vertical velocity is given by

$$v = \frac{1}{2\sqrt{R_0 l_s x}} (\eta f' + f) \tag{1}$$

where

$$\eta = \frac{l_s y}{\sqrt{l_s x / R_0}} \tag{2}$$

and the Blasius equation is given by

$$f''' + 0.5 f f'' = 0 \tag{3}$$

The mistake was in v. The equation should read

$$v = \frac{1}{2\sqrt{R_0 l_s x}} (\eta f' - f) \tag{4}$$

With this, I get results that closely match Bertolotti. I have also computed the mean-flow divergence and it is on the order of 1×10^{-6}. Note that I still have a difference in the peak amplitudes depending on which normalization condition I use. This is particularly concerning since my u_{max} normalized results don't agree perfectly with Bertolotti. Instead, my E_k norm results appear to be better? I think that this error in v may have also been the cause of some of my difficulties in getting a starting solution for the MDF. I might go back and try to update α_{00} now that I have fixed things.

Before doing that I have implemented the normalization condition that Bertolotti uses

$$\frac{\int_0^\infty (\hat{u}^* \hat{u}_{,x}) \ dy}{\int_0^\infty (\hat{u}^* \hat{u}) \ dy} = 0 \tag{5}$$

which is basically just the streamwise component of the disturbance kinetic energy normalization. My results with this are very close to the full E_k normalization and the LPSE results at $F = 86$ seem to match Bertolotti's results well (even though I'm not really clear on which normalization that he used in his figure 5.1. He indicates that u_{max} was used which doesn't agree very well with my results.)

In comparing the *nonlinear* results, it seems that my forcing function was not correct. Upon review, I discovered that I was computing the streamwise derivative of the solution incorrectly. Upon fixing this, my results are in better agreement with Bertolotti's however, they are still not identical? I also have tremendous difficulty in converging the solutions. I think that I will implement the phase locking that Bertolotti uses. In this case, only α_{01} is iterated on. All the rest of the modes wavenumber are integer multiples of the fundamental. I have done runs at lower initial amplitude $A_{01} = 0.1\%$ and convergence is decent (10-12 iterations) and the solutions look qualitatively correct. The number of iterations did not change substantially when I switched between phase locked and non locked. There was a slight difference in the computed α from the non-locked and locked runs. There was also a more significant transient in both α and γ when the wavenumbers (and hence the phase speeds) were not locked. Howver, there was not a significant difference in the amplitude evolution.

It seems that the majority of my difficulties in convergence are due to the formation of higher harmonics. For consistency, I have swithed to Bertolotti's convergence tolerance which is basically looks at the magnitude of the changes in α, *not* the relative changes. Bertolotti does claim that convergence is faster when using trapezoidal. This statement may be misleading – he may be refering to spatial convergence and not convergence of the iteration scheme. I will try converging to 1×10^{-6} to see if the solution is still okay – still seems okay.

Is it possible that aliasing is messing up the results for larger amplitudes. Perhaps I should try to increase the number of temporal modes just to make sure. I increase the number of modes to 24 with no noticable difference.

I have tried to go back and put in the α_{00} term. Now it seems to converge just fine with this in place. However, the solution is somewhat different – presumably more accurate since the exponential growth of the 00 mode is now accounted for exactly. But, the iteration eventually fails to converge.

2 Nonlinear PSE

Note that I initially started these calculations (1-6) using a fundamental that was the result of a previous LPSE run. Because of this, the value of $u_{max} = 0.782248$ which means that to get the actual amplitude, I have scaled the value used in the inflow file by this number.

With a little help from Christopher Hill I was finally able to get the nonlinear PSE working. The problem was completely due to the mean-flow-distortion (MFD) mode which must be handled differently from the other periodic modes. There are two primary difficulties: 1) getting a reasonable starting profile for the MFD and, 2) updating α_{00}. To get a good starting profile, Hill suggests that you solve an inhomogeneous problem for the 00 mode where the forcing function comes from the nonlinear term and you turn off the streamwise derivatives (I was already doing this). The trick is that you have to set α_i for this mode to be slightly negative. Hill suggests that $\alpha_i = -1.0 \times 10^{-3}$ based on δ^*. I find that a more negative value gives better results. Basically you are trying to account for the streamwise derivative using α. However, after the first station, you set $\alpha_{00} = 0$ and absorb all the changes in the MDF profile using the streamwise derivative. In this way, there is no iteration on α_{00}.

For the harmonic, I started a LPSE using the inhomogeneous solution and one using the parallel flow eigenfuction. The resulting $alpha_{02}(x)$ where in excellent agreement except for a slight transient when using the inhomogeneous solution. Note that the eigenfuction caused only a very slight transient. Note, however, that the evolution of this mode is very different in NPSE.

I will try placing all of the initial profiles in inflow files – this worked just fine. I am now trying to compute an initial profile only for the MDF (run2). For the other mode, I do it like I used to. This gives a *much* cleaner solution for the higher mode! So I should only precompute the mode shape for the MDF term. The others can be made on the fly – run2 is my reference solution for low amplitudes.

The initial run was done with $A_{01} = 0.078\%$, this immediately initiated modes $0 - 2$. In run3, I have increased the initial amplitude, based on u_{max} to $A_{01} = 0.78\%$ which initiated modes $0 - 4$. In this run, the MDF was great enough that the stability characteristics of the higher modes was dramatically influence. Similar to Bertolotti, I get that the fundamental continues to grow downstream of the linear Branch II neutral point. Note that the transient associated with the initiation of new modes appears to get worse the higher the harmonic.

In run4, I chose an initial amplitude of $A_{01} = 0.39\%$. This cases a slight increase in instability – something is wrong with the MDF mode in the run? It seems that the MDF mode was messed up in the initial condition. I have rerun this case and have gotten correct results.

In run5, I am investigating the use of different initial conditions for the higher harmonics. I switched back to $A_{01} = 0.078\%$ and used the eigenfunction from parallel theory for the first harmonic with an amplitude of $a_{02} = 4.5 \times 10^{-6}$ which was selected to match the inhomogeneous local solution. Surprisingly, this leads to a larger transient then using the local inhomogeneous solution – perhaps due to inaccurate phase. It looks like it may be best to just use the inhomogeneous solution to initialize the higher harmonics – it certainly is more convenient. In cases where you want more control over the initial condition, you can allways provide the higher harmonics but you have to live with the transient that you get.

In run6, I have increased the initial fundamental amplitude to $A_{01} = 1.56\%$. It indeed does get difficult to converge. I have observed that the higher the initial amplitude, not only does the growth rate increase, but the wavenumber decreases. So, the fundamental

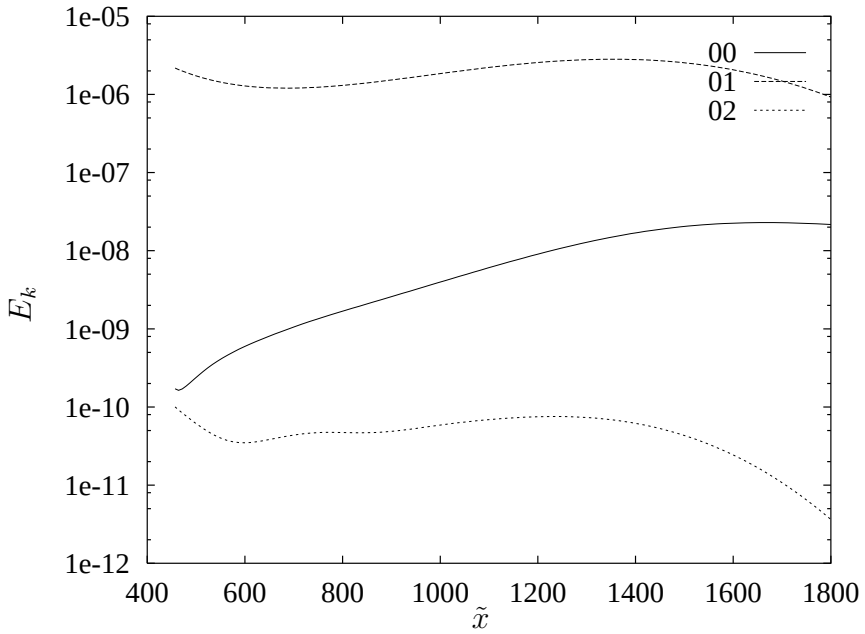

Figure 1: Amplitude evolution from nonlinear PSE: $R_0 = 200$, $F = 150$, $A_{01} = 0.1\%$.

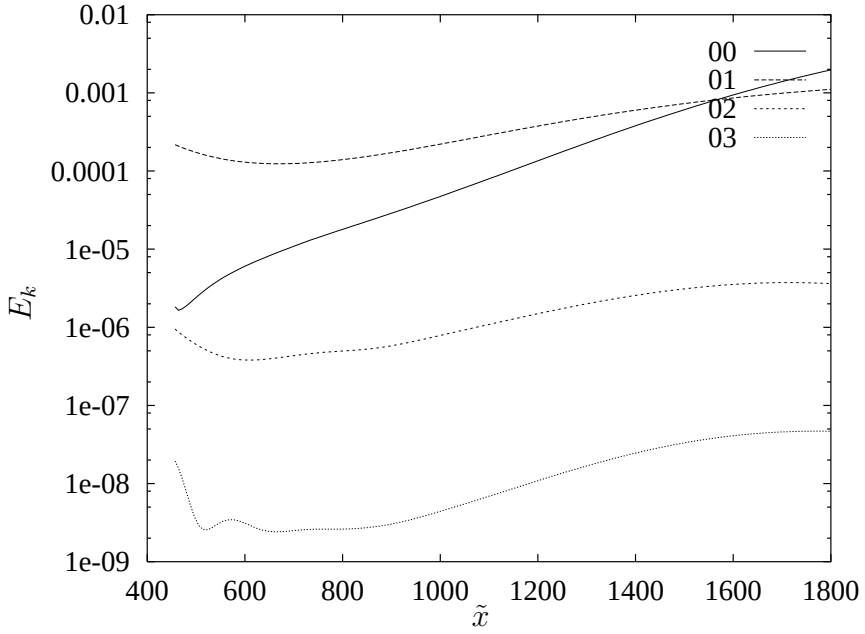

Figure 2: Amplitude evolution from nonlinear PSE: $R_0 = 200$, $F = 150$, $A_{01} = 0.078\%$.

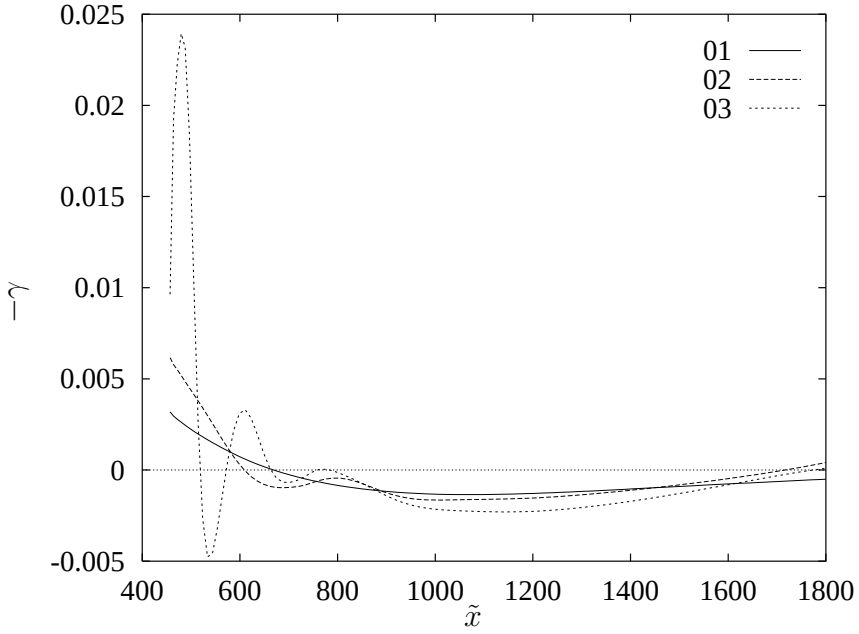

Figure 3: Evolution of modal growth rates from nonlinear PSE: $R_0 = 200$, $F = 150$, $A_{01} = 0.78\%$.

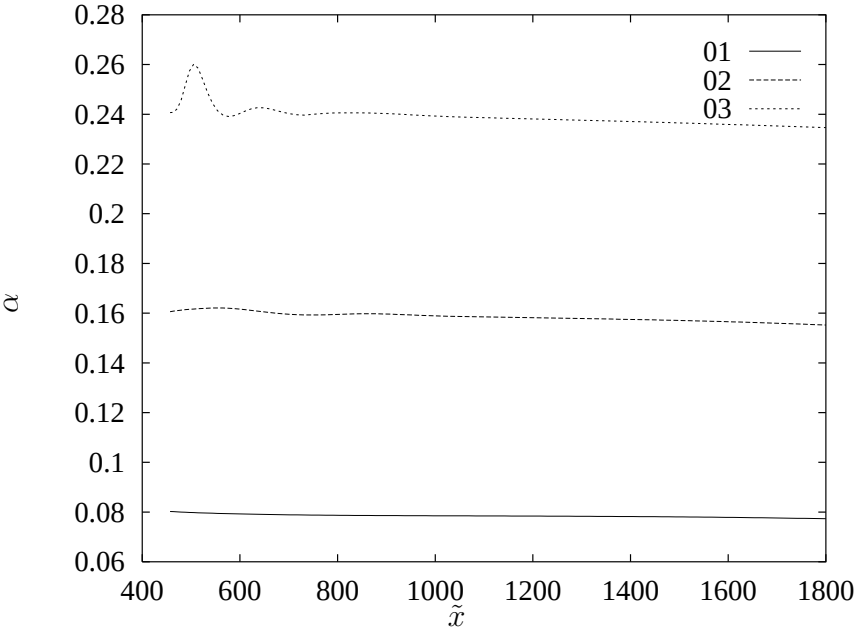

Figure 4: Evolution of wavenumber from nonlinear PSE: $R_0 = 200$, $F = 150$, $A_{01} = 0.78\%$.

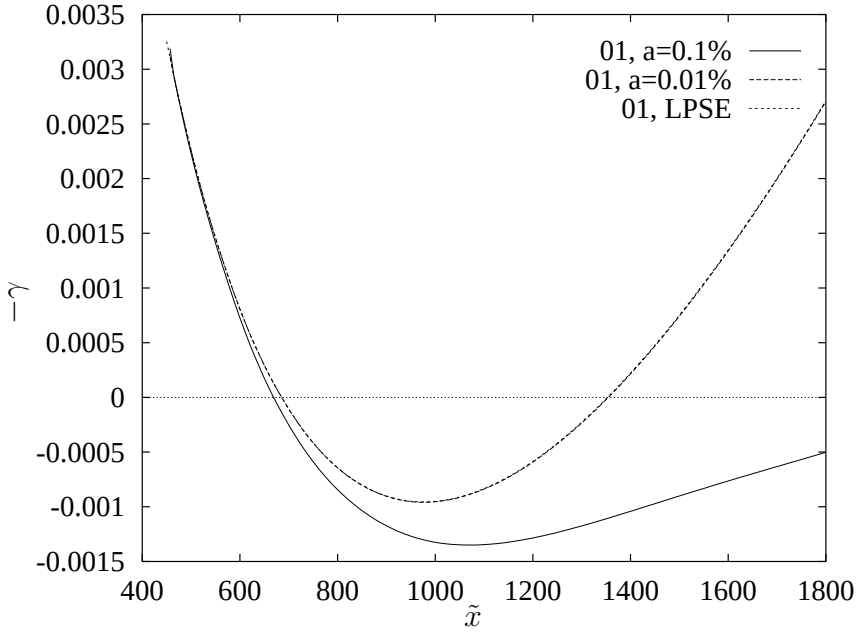

Figure 5: Effect of initial fundamental amplitude on fundamental growth-rate from nonlinear PSE: $R_0 = 200$, $F = 150$.

grows faster and is longer due to nonlinear effects. For this case, the growth rate of the fundamental goes out the roof and eventually blows up.

In run7, I used the LST eigenfunction as the fundamental initial profile using a corrected initial $A_{01} = 0.2\%$ based on u_{max}. As expected the use of the LST eigenfunction causes a transient that is most noticable in the fundamental growth-rate. Compared to my prior runs, this run most closely matches run4 in which the corrected initial amplitude is 0.39.

I think that I will do a run8 in which the initial amplitude of the LST fundamental eigenfunction is set to 0.4 in order to match with run4 above. The runs are in close agreement with the run8 fundamental a little higher, as expected. Interestingly, the transient for the 02 mode is greater for run4 as compared to run8. However, the fundamental is cleaner for run4, especially noticable in the growth-rate.

2.1 Comparison to Bertolotti

Bertolotti starts with $R = 400$ using $F = 86$. He includes six Fourier modes in time and three initial amplitudes: 0.20%, 0.25%, and 0.30%. I have constructed a Blasius field using $Nx = 200$ from $R = 400$ to $R = 1000$ with $R_0 = 400$.

3 Discrete Adjoint Solution

1. It took quite a bit of effort to get a clean discrete adjoint method

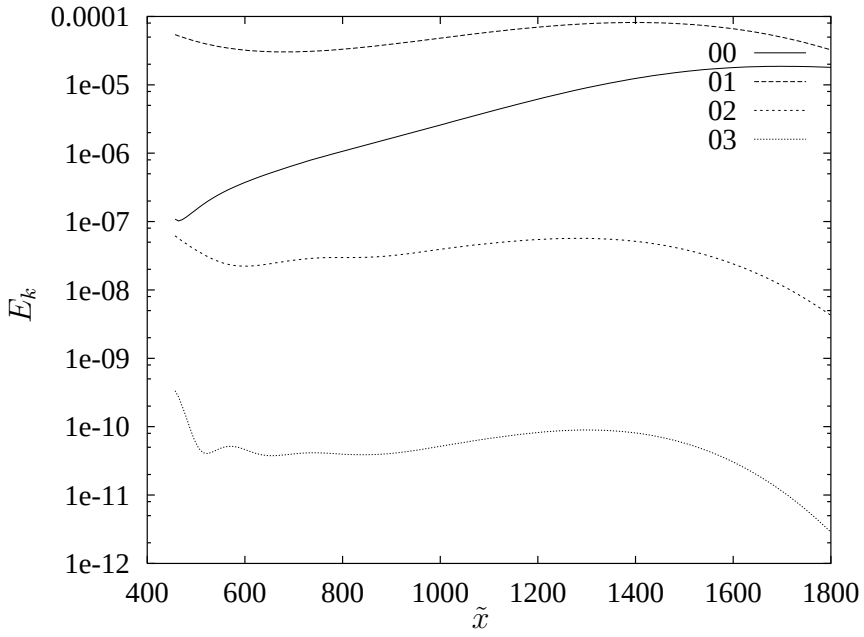

Figure 6: Amplitude evolution from nonlinear PSE: $R_0 = 200$, $F = 150$, $A_{01} = 0.4\%$.

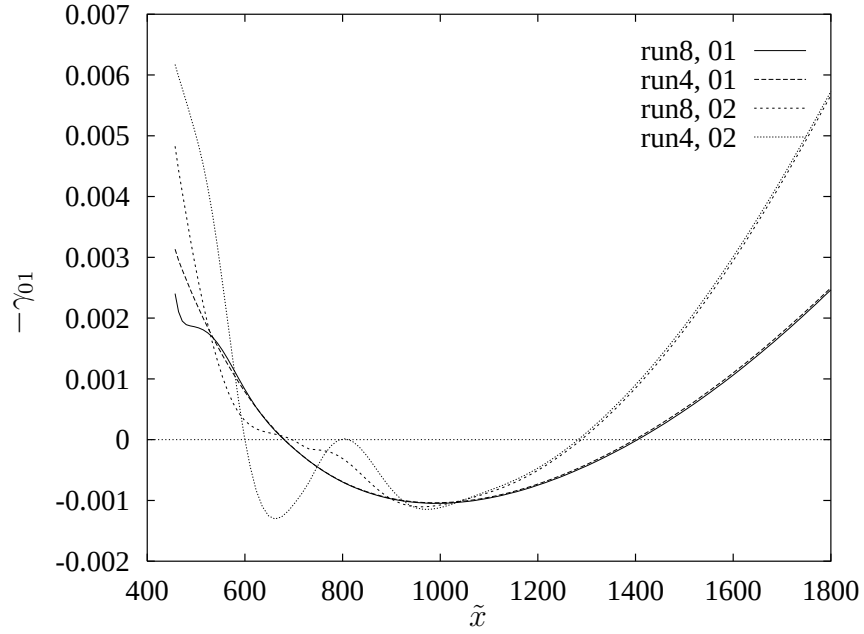

Figure 7: Comparison of fundamental growth-rates from nonlinear PSE: $R_0 = 200$, $F = 150$, $A_{01} \approx 0.4\%$.

2. The main impediment was that I didn't account for the mapping implied by the Chebyschev method! However, this mapping does introduce a problem in that the weight function of the Chebyshev polynomials is given by $w(x) = (1-\eta^2)^{-1/2}$ where $\eta \in [-1, 1]$. Note that the weight goes to zero at the boundaries which when included in the discrete inner product means that the boundary values must be zero. However, the surface pressure is not zero and this leads to a problem. I corrected it by using an artificial, nonzero weight on the wall boundary.

3. With the above fix, the discrete solution is smooth in all quantities except for approximately 5 nodes in pressure near the wall when using the $\partial v/\partial y = 0$ wall BC. The error in the wall pressure appears to be about 4%.

4. I went to great lengths to statically elliminate the Neumann boundary conditions. If you don't do this the discrete adjoint wall pressure is 0 when using both the $\partial v/\partial y = 0$ wall BC and the wall normal momentum equation.

5. If I use the wall normal momentum equation to determine the wall pressure, there is a significant oscillation in v and the pressure also is in error. This is better if I set $\partial v/\partial y = 0$, although there is still a small error in the wall pressure. Again, things are better if you statically elliminate the Neumann boundary condition.

6. I do think that a lot of these problems are coming from the zero boundary weights for Chebyschev.

7. The finite difference discrete adjoint does not work as well. Although the mapping in this case remains finite at the boundaries. My guess is that I am seeing the classic problem of an overconstrained pressure. If I were to use a staggered mesh, I would anticipate that many if not all of these problems would be reduced.

8. The bottomline is that I can compute the discrete adjoint using the spectral method. The eigenvalues are identical to the regular problem and the eigenfunctions should be highly orthogonal.

9. There is a tendency for the discrete adjoint solutions to exhibit more node-to-node oscillation than the regular equations. I think that this is a property (undesireable) of the discrete opperators used.

10. I had been using the y-parameter $y_{max} = 80$, $y_{str} = 0.01$. This stretching is too much – it causes oscillations in the mean flow derivative. I backed off to $y_{str} = 0.05$ and things look good. The adjoint may not be as good since the gradients near the wall are higher. Now try $y_{str} = 0.025$ – this is better for the adjoint.

4 Linear PSE validation

I have revised the `blasius.f90` to base the input and output on the Blasius length scale $\tilde{\delta}_0 = \sqrt{\tilde{\nu}\tilde{x}_0/\tilde{U}_e}$ so that the reference Reynolds number is given by $R_0 = \tilde{U}_e\tilde{\delta}_0/\tilde{\nu}$. To get things in terms of the displacement thickness you need to use the relationship $\delta^*/\delta = 1.7208$. The frequency parameter is defined as

$$F = \frac{\tilde{\omega}\tilde{\delta}_0}{\tilde{U}_e} 10^6/R_0 \tag{6}$$

Note that the nondimensional inputs to the PSE code are R_0 and $\omega = FR_0 10^{-6}$. In this case the reference length-scale is $\tilde{\delta}_0$, the velocity scale is $\tilde{U}_e$ and the time-scale is $\tilde{\delta}_0/\tilde{U}_e$. If you want to convert the x coordinate back to the local Blasius coordinate, then you multiply by the quantity $\sqrt{R_0/x}$ which I typically do in an `awk` script.

To test the code I have run the case $F = 150$ from $R = 300$ to $R = 600$. From my PSE run I get the first neutral point at $R_I = 370$ and the second at $R_{II} = 520$.

The growth rate is defined as

$$\gamma = \frac{1}{Q}\frac{\partial Q}{\partial x} \tag{7}$$

Bertolotti defines the following notation: u, v are real, physical velocity components, u', v' are the root-mean-square of physics velocities, and $\hat{u}, \hat{v}$ are the complex velocity profiles. He bases his normalization criteria on u'_{max} where the maximum is taken in the wall normal direction. He says that the amplitude based on u'_{max} is proportional to the vorticity strength in the critial layer making this an important quantity for secondary stability analysis. The complex wavenumber[1] a is then given by

$$a(x) = -i\frac{1}{\hat{u}(x, y_m)}\frac{\partial \hat{u}(x, y_m)}{\partial x} \tag{8}$$

so that the growth-rate $\gamma = -Im(a)$ and the wavenumber is $\alpha = Re(a)$.

Bertolotti always set $\partial a/\partial x = 0$. I find that there is no advantage to doing so.

I have implemented a second-order accurate time advancement scheme based on midpoint rule. The convergence is a little slower and it does poorly in the transient – 2-Δ waves are formed. If I first start with backward Euler and then switch to midpoint the solution is good. However, for the limited resolution studies that I have done, there is no advantage of using a second order scheme – even with only 4 points per wave. However, if you intend to resolve the transient then there could be an advantage to a second order scheme. However, you really have to be careful since I don't think that the transient with PSE is accurate anyway! Note that Bertolotti starts with Backward Euler and then smoothly changes to Trapezoidal and that he finds that with Trapezoidal a converges faster. I do not observe a speedup in convergence with trapezoidal although I haven't tried to smoothly change to it.

[1] Note that Bertolotti's a is equal to i times mine.

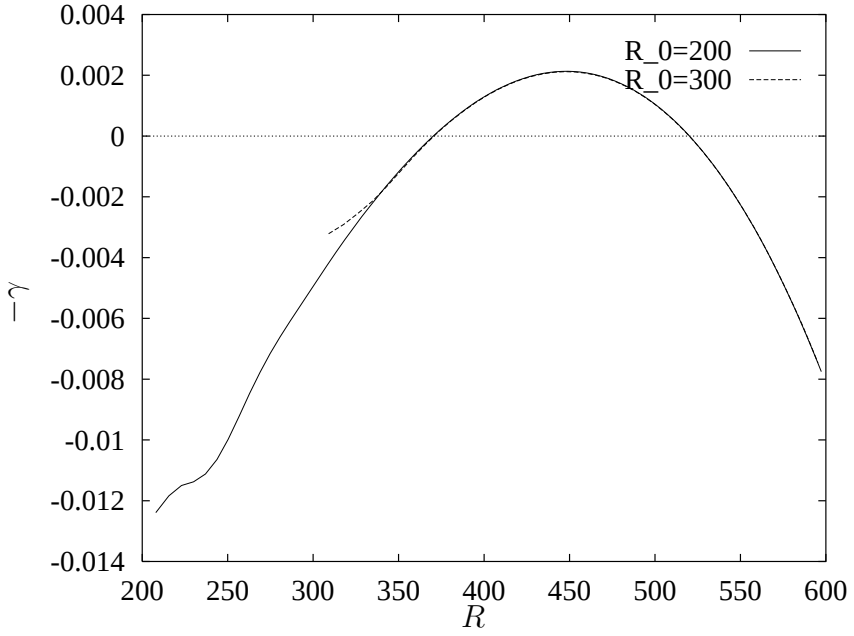

Figure 8: PSE growth rate based on disturbance kinetic energy for Blasius boundary layer at $F = 150$.

I have experimented with the wall pressure boundary condition. It seems like the best alternative is to enforce continuity at the wall. If you use the wall normal momentum equation there can be a pressure jump at the wall. In body-fitted coordinates the continuity equation is given by

$$c_1 u_{,x} + c_2 v + v_{,y} + w_{,z} = 0 \tag{9}$$

where

$$c_1 = \frac{1}{1+\kappa y} \qquad c_2 = \kappa c_1 \tag{10}$$

and κ is the local surface curvature. Typically due to the no-slip condition $u_{,x}$ and $w_{,z}$ will be zero. However, if one uses a linearized roughness boundary condition this will not be the case. For generality, these terms are included when solving for v at the wall using this approach.

I have implemented two different strategies for normalizing the PSE shape funtion. The figure is as used by Bertolotti

$$\frac{1}{\hat{u}_{max}} \frac{\partial \hat{u}_{max}}{\partial x} = 0 \tag{11}$$

The second is from Herbert

$$\frac{\int_0^\infty \left(\hat{u}^* \hat{u}_{,x} + \hat{v}^* \hat{v}_{,x} + \hat{w}^* \hat{w}_{,x}\right)\ dy}{\int_0^\infty \left(\hat{u}^* \hat{u} + \hat{v}^* \hat{v} + \hat{w}^* \hat{w}\right)\ dy} = 0 \tag{12}$$

Overall the code works better with the second normalization. However, there are some differences in the growth-rate results using both codes. Futhermore, the wavenumber tends

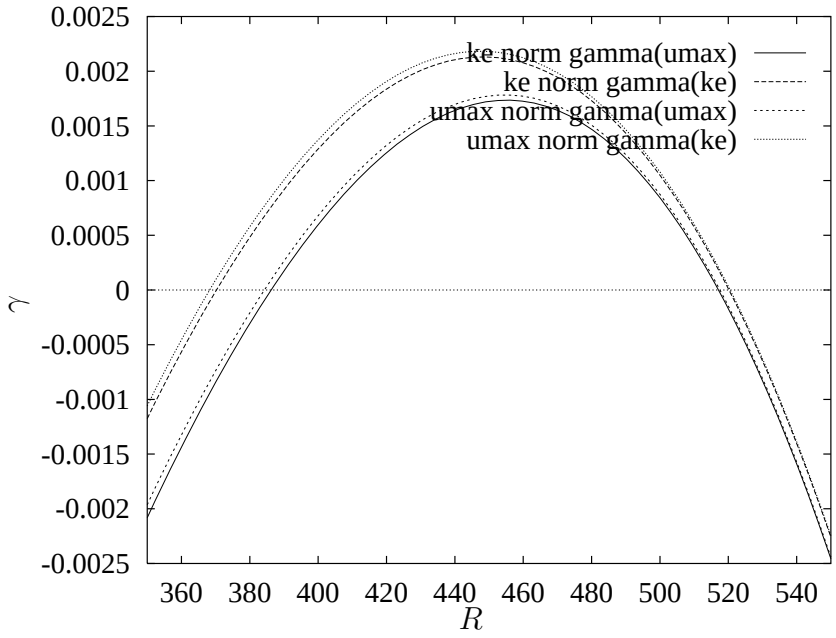

Figure 9: Comparison of PSE growth-rate results based on E and $\hat{u}_{max}$ for Blasius boundary layer at $F = 150$.

to get corrupted when using the u_{max} normalization. This may be due to the rather poor method that I am currently using to find the maximum. I have implemented a B-Spline method for determining the maximum which should be much more accurate. This has fixed the oscillations in a that I had previously observed.

However, I still get slightly different growth rates when using the different normalizations. Perhaps the transient is worse when using the u_{max} normalization. As the R increases the differences between the two normalizations vanishes. I have performed a run starting at $R_0 = 200$ still at $F = 150$ and (as seen in figure 8) it looks like the differences are *not* due to the transient but are an artifact of the normalization. Note that to plot the code results on a constent scale, the x coordinate must be multiplied by $\sqrt{R_0/x}$ and the complex wavenumber by $\sqrt{x/R_0}$. Figure 9 shows the effect of PSE normalization on the predicted growth-rate using two measures. I think that it would be a worth performing a comparison of LNS and PSE results to determine which of the methods is the best.

5 Comparison to HLNS

Take the $R_0 = 300$ to $R = 600$ case and perform a forced HLNS solve on it. In order to do this, I need to make a new x-mesh in `blasius.f90` so that there is refinement near the inflow. I am going to use the same technique used by Streett in tt hdir.f90. The input parameters are x_1, x_2, and Δx_1. Basically, x_1 and x_2 are breakpoints. The mesh begins with spacing Δx_1 and is uniform till x_1. The mesh then transforms to a new spacing between x_1

and x_2 and is uniform at the new spacing after x_2. Between x_1 and x_2 the mesh spacing changes quadratically.

I have modified `blasius` to support the nonuniform x-mesh. With the new mesh, I have rerun PSE and have verified that the results are mesh independent. Now use the inflow data and mean flow to compute a HLNS solution. In `hdir` I implemented a new inflow boundary condition `ibci=3` that reads an eigenfunction file `inflow.dat` and places that complex profile on the inflow boundary. My first HLNS is on the same mesh as the PSE so that it is very poorly resolved. I had an error in the inflow boundary condition which has now been fixed. Okay, things appear to be working now. First, check the growth-rate computation and compare against PSE results. It is of the same magnitude, but considerably more oscillatory.

I have made several HLNS runs on a $Ny = 32$ mesh to experiment with streamwise resolution. The domain covers the same range as that in the PSE calculation. I am currently unsure as to whether the buffer domain is working properly. It appears that I am getting some significant reflections? I have tried runs with $Nx = 200, 400, 800, 1000$ to see the effect of streamwise resolution. Things have gotten better with $Nx = 100$ although I still have significant oscillations in my HLNS growth rate. I bet that these are due to poor resolution in the buffer domain so I have increased the resolution to $Nx = 1500$ to test this hypothesis. I may want to try the damping function used by Fasel which they claim reduces reflections. The results with $Nx = 1500$ are virtually the same as the $Nx = 1000$ case. I am now trying to increase $amuc = 15$ instead of 8. It is virtually identical! Now trying an increase of amu2c.

run1, amuc = 8, amu2c=20 run2, amuc = 15, amu2c=20 run3, amuc = 15, amu2c=80

Again, this gives no significant improvement!

Looking at the latest fields shows that they are relatively clean. I might be able to improve the results by refining near the inflow boundary since there is a significant transient there. All previous runs where performed with $fx1 = 0.5$ and $bx1 = 0.2$. I am switching this to $fx1 = 0.25$ and $bx1 = 0.2$ to evaluated the effect of inflow resolution – no significant difference.

On Howard's airfoil, I ended up using $amuc = 6$, $amu2c = 20$ and the results there were *much* better! I'll give these a shot here. Note that I used very high resolution in y for Howard's airfoil.

Now try increasing vertical resolution from $Ny = 32$ to $Ny = 48$ to see if this has an influence. The results seem to be better up to $x = 1000$ but then get worse – probably due to my buffer layer (this observation is for the growth-rate based on u_{max}, for dke the y resolution made no significant change. Switching back to $amuc = 6$ and $amu2c = 20$ just like that used for Howard. The solution is not really any difference. How about making the buffer domain a little longer – try changing *amuloc* from 1600 to 1400. With the longer buffer the results are basically the same.

The final thing that I can try is to use a PSE profile on the inflow. This should dramatically reduce the initial transient. For example, I could put a profile from the $R = 200$ run onto the inflow of the $R = 300$ mesh. If the oscillations are an artifact of the inflow forcing,

then I should be able to reduce them. In the case of Howard's airfoil, I was doing receptivity simultions which perhaps lead to less oscillations in the growth-rate. I could try placing a suction/blowing slot at, say, $b_{loc} = 400$ with a $b_{con} = 100$. This should be fairly well resolved on the current mesh. It could be the case that the parallel-flow eigenfunction is such a poor approximation of the nonparallel flow natural mode that it excites a host of poorly damped modes. This may particularly the case since the inflow forcing occurs for all y.

Trying suction/blowing for the $Ny = 32$ case.

Things work much better for Howard's airfoil case? Return to this case and verify that I get the same answer. I have run this case on the SGI and I get results that are identical to the original Cray version.

I have restructured the Blasius run based on the successful Howard run. This includes: increasing y_{max}, making the buffer $2\lambda_{ts}$, using the same buffer parameters, increasing $Ny = 64$, using 40 nodes per streamwise wavelength where $\lambda_{ts} \approx 80$. In the domain from 200 to 1800 there are approximately 20 periods so I am using 800 points. Note that on Howard's airfoil, I really only had about 15 points per wavelength. For this same resolution, I could get away with 300 points in the Blasius case.

In order to make a high signal-to-noise ratio, I have used a suction/blowing source at 400 with width of 40 units. This is roughly similar to Howard's case except that relative to Branch I, the source location is further upstream. If you want the maximum effective amplitude then I should really put the source near Branch I (which is at approximately $x = 680$). However, there has to be some projection onto the most unstable mode so that I should get a reasonable response. Note that at this F there are only about 8 wavelengths of the instability wave within the unstable region.

If you really want to get a low transient solution, then I guess that you had better put a PSE shapefunction on the domain inflow. If I start PSE at $R_0 = 200$ then I can extract the profile at $x = 450$ an use it in a calculation starting at $R_0 = 300$.

I have implemented the buffer function used by Meitz & Fasel which is defined by

$$c(\xi) = \exp\left(-\frac{\xi^4}{10}\right)\left(1 - \xi^5 0\right)^4 \tag{13}$$

where $\xi = (x - x_B)/(x_{max} - x_B)$. They claim that this buffer function reduces the relection of waves from the junction at the upstream end of the buffer domain. If it works, this seems like a good idea since I am having extreme resolution difficulties due to reflections at the junction. The first run that I have done with this buffer is for the $Ny = 32$ case with boundary forcing since I have data to compare against using the original buffer. The results look identical to the original buffer function. However, I guess that I should try to do a calculation with fewer points in x to see if the new function is easier to resolve. To test this, I am trying $Nx = 300$ – the results seem no better than what I had with the old buffer function. Now trying $Nx = 600$ – this is better but still not as good as the $Nx = 1000$ case. From 1000 to 1500 the difference is small.

It really seems to boil down to signal-to-noise ratio. It appears that you may have a problem when the amplitude of the instability wave is small. For example, if you force at the

boundary far upstream of the NP then the primary instability wave may be too small relative to errors caused by the buffer-layer for an accurate measure of the growth-rate. I'm going back to the $R_0 = 300$ case and placing a PSE shape-function on the inflow boundary. To get the inflow condition I ran a PSE run starting at $R_0 = 200$ to $x = 450$. I then modified the `blasius.f90` code to allow for an arbitrary R_1 given a reference R_0. Thus, I keep $R_0 = 200$ but set $R_1 = 300$. Of course the resolution for the HLNS run is increased in x and I used the same outflow boundary location $R_2 = 600$.

1. The result is fairly good agreement with PSE with very little inflow transient, but I still get oscillations in the growth rate. Originally I set $x_b = 1600$.

2. On a second run I am setting $x_b = 1500$ to see if a more gentle buffer transition will help reduce oscillations – no difference.

3. Try increaseing $amu2c = 200$ on run3 – basically no difference in the growth-rate, although the influence of the buffer is felt further upstream.

4. Now trying $amuc = 0$ on run4 – this actually worked fiarly well, although there was a buildup of energy at the outflow, the upstream influence was similar. The energy build-up may be due to the pressure specification. Otherwise, it looks like the wave progresses fiarly smoothly out the domain.

5. It looks like the main problem is due to the specification of pressure on the outflow boundary at $Ny - 1$. To investigate this, run5 sets the pressure at $Ny/2 + 1$. The result is generally the same, but with some 2Δ wave in the solution.

6. Going back to setting pressure at $Ny - 1$, try using $ivbc = 2$. Using continuity at y_{max} gives considerably better results as shown in figures 10 and 11. This indicates that the trouble is likely due to the upper boundary condition. Right now, the upper boundary is rather close to the wall at $Ny = 120$.

7. I think that the best solution is to move the upper boundary farther away from the wall. Try $y_{max} = 200$, $y_{str} = 0.08$ on run7. By moving the upper boundary further away, the error due to setting the pressure to zero at the corner point should be reduced. I am also using continuity on the upper boundary which was shown in run6 to give improved results. The taller mesh does give improved results as seen in figures 12 and 13, although I think that there is a trade-off between resolution and boundary location.

8. Now try $N_y = 400$, $y_{str} = 0.05$. Things get worse. I guess that the y-resolution is not good enough.

9. Try increasing $N_y = 48$ to see if the problem is resolution related. This didn't fix the problem. It seems that the problem is due to the floating mean pressure. Fixing the pressure on the outflow does not seem to constrain the upstream pressure to the correct value. In fact, the inflow pressure at y_{max} is nowhere hear zero. Instead it is a complex constant approximately equal to $-0.12 - 0.56i$. Setting the pressure only

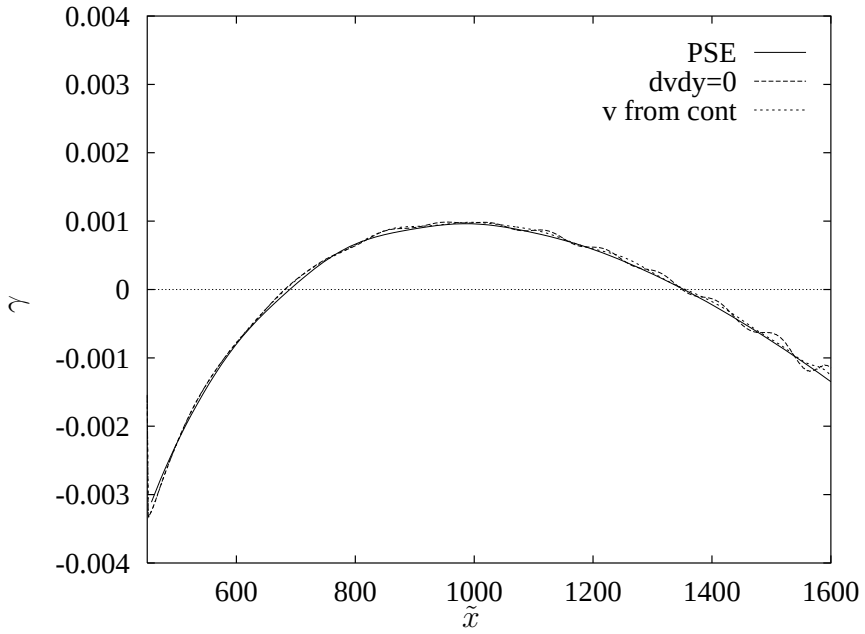

Figure 10: Comparison of PSE growth-rate results based on E for Blasius boundary layer at $F = 150$.

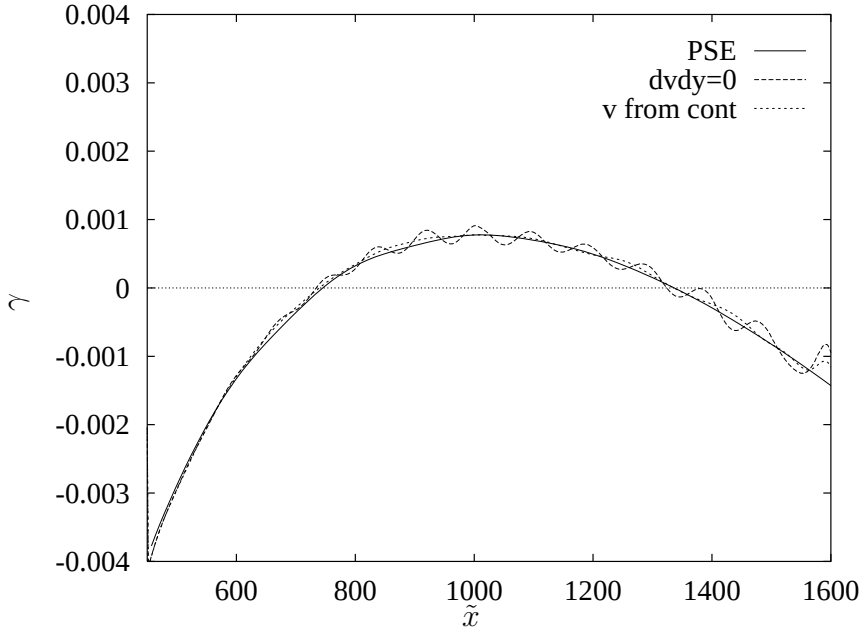

Figure 11: Comparison of PSE growth-rate results based on u_{max} for Blasius boundary layer at $F = 150$.

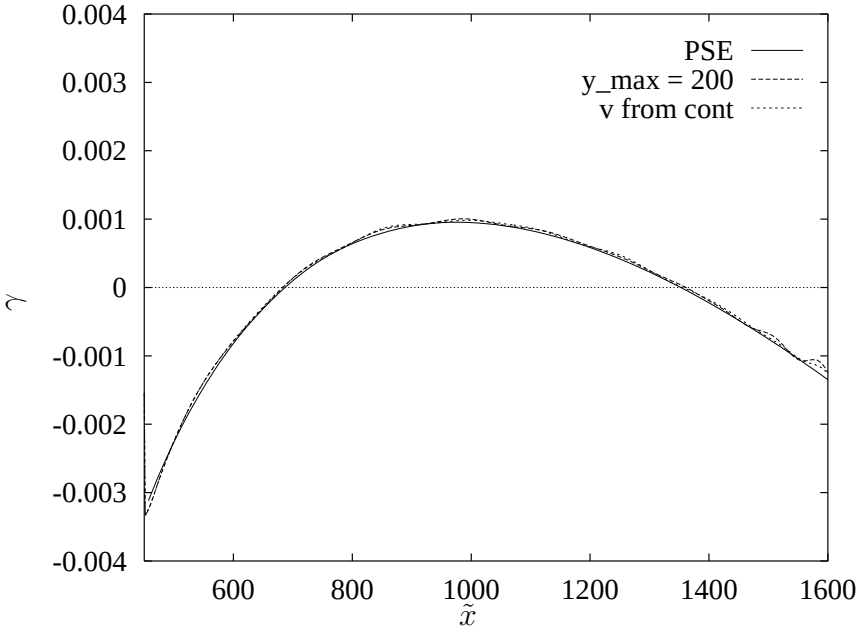

Figure 12: Comparison of PSE growth-rate results based on E for Blasius boundary layer at $F = 150$.

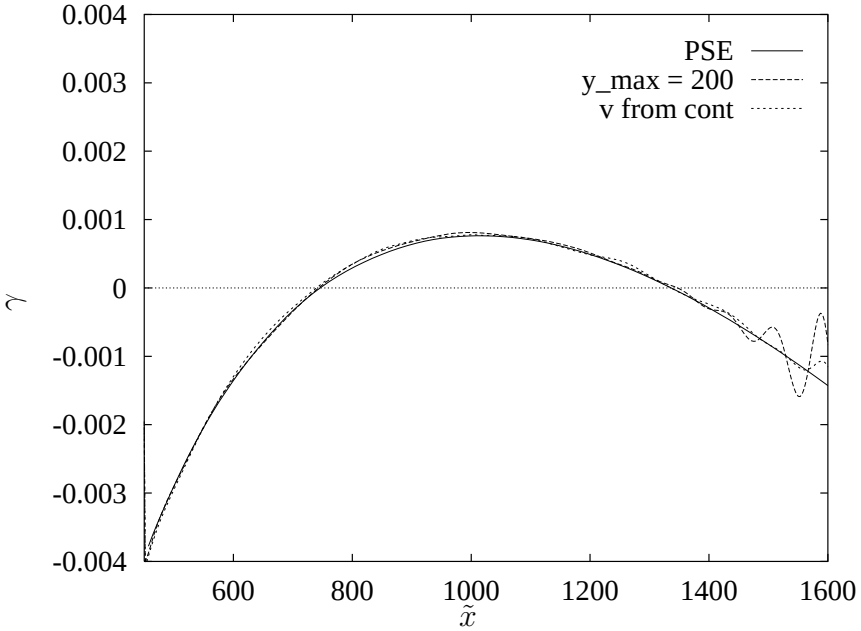

Figure 13: Comparison of PSE growth-rate results based on u_{max} for Blasius boundary layer at $F = 150$.

on the inflow didn't work. Now I'm trying setting pressure to zero all along the top boundary. This should be a good approximation... Maybe the buffer does generate a pressure gradient which is why the pressure cannot be constant in the far-field?

10. Okay, I am setting the pressure to zero on the inflow. Using $\partial p/\partial y = 0$ on the interior and outflow and am now using continuity on the ouflow.

It does look like the Fasel buffer function may work better than Streett's. I tried to turn on a little bit of $amu2c = 1$ and the results are worse than without it. What if I arbitrarily increase the amplitude of the inflow wave. Since everything is linear – this should cause no difference in the results. I may also want to increase the x resolution just to make sure that is the culprit. Right now, the HLNS amplitude is a little high – usually this occurs when the x-resolution is low. I increased $N_x = 1000$ to see whether the amplitude and growth-rate converge to the PSE values. There was no significant difference in the solution.

If you ramp down the streamwise viscous terms in-order to make the equations parabolic near the outflow, it seems to me that $\partial p/\partial x$ must increase in the buffer to compensate. The other option is to increase the y-viscous terms so that there is a new balance. Of course, doing so will be quite tricky in terms of setting the factor on these terms. If you could estimate $\partial^2 u/\partial x^2$ then you could just add this term on as a source term. This is possible if you know an approximate value for α, say from PSE analysis. Given α which is assumed to be constant, then you can write

$$u = \hat{u} e^{i\alpha x} \tag{14}$$

then the first derivative is given by

$$u_{,x} = \hat{u}_{,x} e^{i\alpha x} + i\alpha \hat{u} e^{i\alpha x} \tag{15}$$

which can be solved for

$$\hat{u}_{,x} e^{i\alpha x} = u_{,x} - i\alpha \hat{u} e^{i\alpha x} \tag{16}$$

The second derivative, after using the above expression, is then given by

$$u_{,xx} = \hat{u}_{,xx} e^{i\alpha x} + 2i\alpha u_{,x} + \alpha^2 u \tag{17}$$

Thus, if $\hat{u}_{,xx}$ is assumed small, then the second derivative can be approximated by

$$u_{,xx} \approx 2i\alpha u_{,x} + \alpha^2 u \tag{18}$$

I have used this in the code and it really doesn't do much. I don't understand why this doesn't fix the problem. I guess that I could include the actual distribution for α, but I really don't think that this will help. By making the buffer a little longer $x_b = 1400$ instead of $x_b = 1500$, I was able to improve the results slightly, but the extent of the upstream influence did not change.

Eureka, the problem is in the streamwise pressure gradient, of course! You have to do the following

$$p_{,x} = \hat{p}_{,x} e^{i\alpha x} + i\alpha p \tag{19}$$

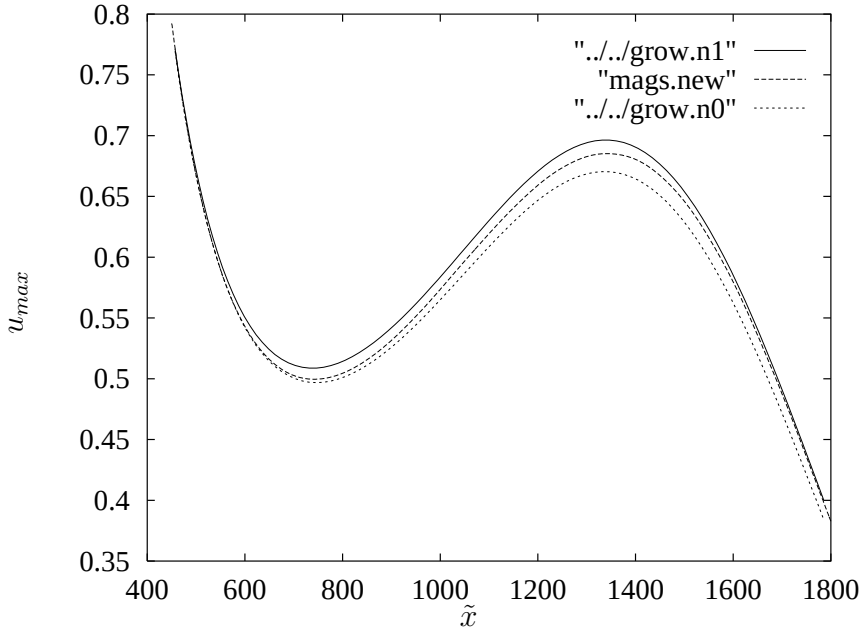

Figure 14: Comparison of amplitudes based on u_{max} for the Blasius boundary layer at $F = 150$.

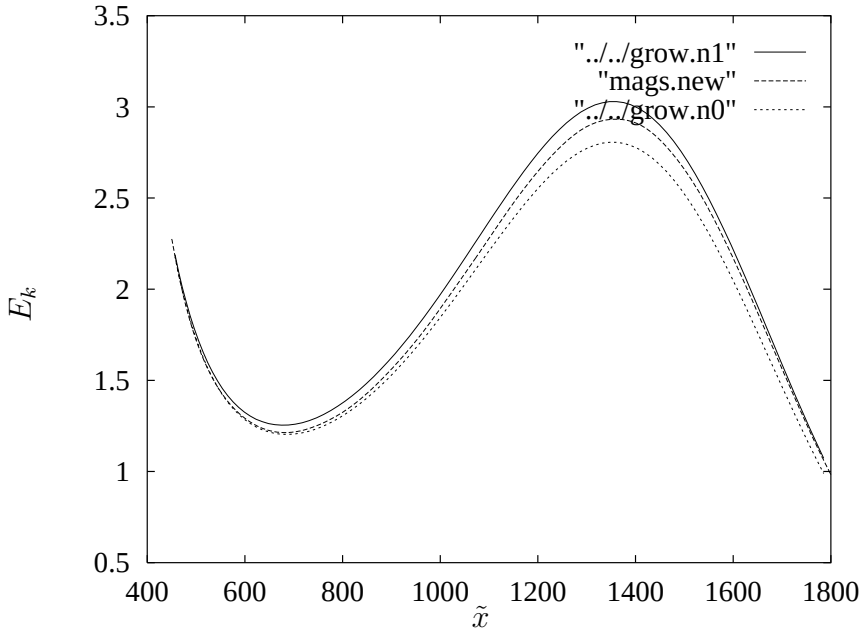

Figure 15: Comparison of amplitudes based on E_k for Blasius boundary layer at $F = 150$.

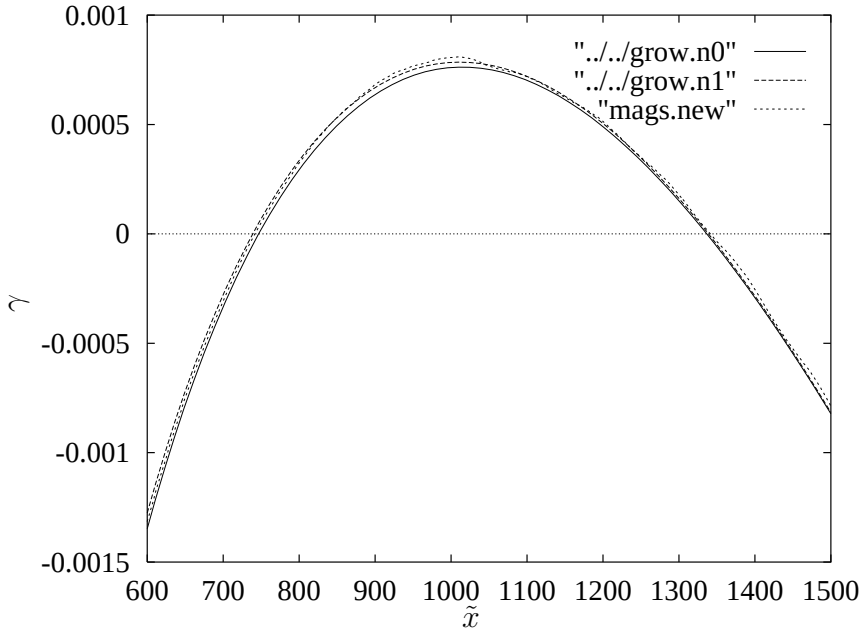

Figure 16: Comparison of growth-rate based on u_{max} for the Blasius boundary layer at $F = 150$.

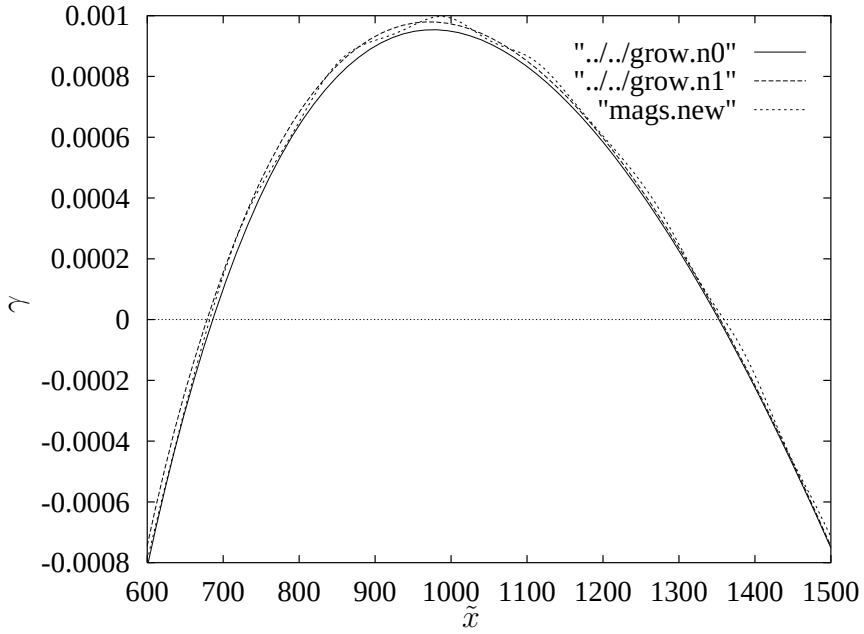

Figure 17: Comparison of growth-rate based on E_k for Blasius boundary layer at $F = 150$.

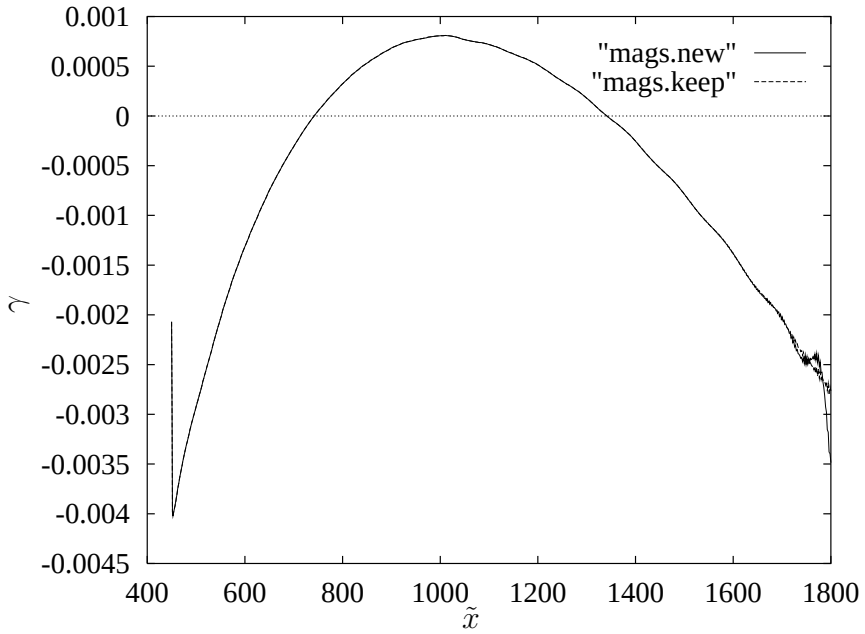

Figure 18: Comparison of growth-rate based on u_{max} for the Blasius boundary layer at $F = 150$: —— real α, ---- is complex α.

To make the PSE parabolic, the term $\hat{p}_{,x}$ is assumed to be zero. If I do the same thing in the HLNS code, **I get an almost perfect outflow boundary condition!!!**

The current results shown in figures 14 – 17 deomonstrate the success of this boundary treatment. Note that in these results I have used the value of $\alpha = 0.07820 + i0.000711024$ which is the PSE predicted value at the outflow (using KE normalization). It seems to me, that the critical feature is to have the real part – the imaginary part is nearly zero anyway. I have tried a new run using on the real part and the results are shown in figure 18 compared to the full complex α. Clearly the results are not quite as good, although they are still superior to the original buffer and the errors are localized near the boundary. Since it is relatively easy to estimate the real part of α this may be a useful method.

- Is it neccessary to include the viscous terms? Maybe all I really need to do is approximate the pressure gradient?
- Will the solution quality be improved if I use the exact PSE $\alpha(x)$ distribution? I tried this and the solution is virtually identical to using just the outflow value of α.

If you set $amu = 0$ then the HLNS code will really solve the PSE since the second derivatives in the streamwise direction have been removed and the pressure gradient is only given by the wave component. However, it is a little bit different since I am still requiring that u represent the total variation of the solution. Therefore, I will still need sufficient resolution to resolve the wave component of the solution. The next step is to code the equations similar to PSE so that the solution $\hat{u}$ is on the slowly varying shape-function.

Then I should get very similar results to PSE. However, the current method, up to trunction error, should be similar to PSE since the same assumptions are made. There should be only slight upstream influence since both the pressure and viscous terms are parabolized.

This works very well, although there is a slight increase in 2Δ that is noticable in the growth-rates. This is probably due to the difference in numerical methods/accuracy between the two methods and the use of central differencing in the HLNS code. The upshot of this is that the HLNS results with forced α are identical to the PSE results. In other words, the full LNS solution shows greater growth than the PSE. It might be possible that this growth is a result of a transient, but I don't think so. The HLNS growth-rates still have a slight oscillation to it, but is not at the same mean value as the PSE. There is a slight error in pressure on the inflow boundary when using $\alpha(x)$ in the HLNS. I'm not sure of the source of this, but it doens't seem to affect the rest of the solution.

6 Re-Validation of PSE/HLNS

Using the corrected mean flow results and the new outflow buffer technique, I quickly reran the PSE/HLNS comparisons that I had made previously at $F = 150$. These runs use the PSE shapefunction inflow profile. It appears that the solutions are in good agreement between the methods, although the general conclusion is the same as above – the growth rate with HLNS is slightly higher than PSE. I also tried a quick resolution study in x to make sure that $N_x = 800$ is adequate. I tried $N_x = 600$ and the transient after the inflow is stronger and the 2δ waves at the outflow are larger in amplitude. For $N_x = 1000$, the inflow transient is better than the $N_x = 800$ case, and the 2Δ near the outflow are smaller. However, the growth-rate is still larger with HLNS compared to PSE. With $N_x = 1200$ there is no decernible difference with the $N_x = 1000$ case, including the growth rate curves. The overall quality of the results is very good, however there still is a slight transient in the HLNS growth rate, especially the growth rate based on disturbance kinitic-energy.

7 PSE validation revisited (8-4-99)

1. As a sanity check, I have gone back to the parallel flow case to validate my PSE and APSE. First I looked at LST and compared continuous and discrete PSE eigenfunction and they are in excellent agreement.

2. I did a 3d TS wave on a parallel Blasius boundary layer and LNS and PSE are in perfect agreement.

3. I have returned to the Parabolic cylinder and have compared PSE and LNS for $\beta = 35$. There are some differences which could be due to transients! Otherwise the results are in good agreement. It is very difficult to get good inflow profiles that are transient free for PSE. This makes it very difficult to get clean growth rate curves near the upstream neutral point. For the bump case, I tried starting a PSE at $i = 227$. This gives very

good results, although the growth-rate predicted by PSE is a little larger than that from LNS, especially near the location of maximum growthrate. Could the LNS need more resolution? This should be established.

4. What if your run APSE backwards, get the adjoint shapefunction and use this to project to a good PSE shapefunction? Try this for the `pcyl_new/beta=100` case.

5. The new buffer works well for crossflow instabilities on the parabolic cylinder. I tried `amuc` of 8 and 20 with no discernable difference.

6. I have compared all the HLNS codes and they are in good order.

7. I have compared the `src` and `dev` versions of the PSE codes and they are also in good order. The main differences lie in the solvers `asolver.f90, lst.f90, tlst.f90, dasolver.f90` and `pse.f90`. Currently the adjoint solver in `src` uses the KE normalization which is clearly wrong. The adjoint solver in `dev` uses the PSE alpha and does one solve to get the shape functions. This is less wrong, but still not right of course. In this case $\alpha = -\tilde{\alpha}$ but $\hat{\boldsymbol{j}}_{,x} \neq 0$.

8 Adjoint PSE

1. I have implemented an iteration into the `asolver.f90` code in `dev`. This uses the following normalization for

9 Nonlinear HLNS

10 Boundary conditions

Currently I use the following far-field boundary condition for LST: $u = v = w = p = 0$. I think that a more reasonable condition would be zero traction. To set this BC requires that

$$u_{,y} = w_{,y} = 0$$

$$p - \frac{1}{Re} v_{,y} = 0$$

I implemented this by requiring that $p = 0$ and $v_{,y} = 0$. This works fine for LST.