Linear eigenvalue code for edge plasma in full tokamak x-point geometry

D. A. Baver and J. R. Myra,

Lodestar Research Corporation,

M.V. Umansky,

Lawrence Livermore National Laboratory

August 2010 (revised May 2011)

submitted to *J. Comp. Phys.*

DOE/ER/84718 LRC-10-137

LODESTAR RESEARCH CORPORATION

2400 Central Avenue

Boulder, Colorado 80301

Linear eigenvalue code for edge plasma in full tokamak x-point geometry

D. A. Baver and J. R. Myra

Lodestar Research Corporation, Boulder Colorado 80301

M. V. Umansky

Lawrence Livermore National Laboratory

A new code is presented for solving linear eigenvalue problems from fluid models of the edge plasma of tokamaks. The 2DX code solves linearized fluid equations in a 2D cross-section of the plasma, with toroidal mode number resolving the third dimension. Geometry capabilities include both closed and open field lines, allowing solution of x-point problems as well as a variety of other toroidal and cylindrical systems. The code generates a pair of sparse matrices forming a generalized eigenvalue problem which is then solved using a standard sparse eigensolver package. Use of a specialized equation parser permits a high degree of flexibility in both equations and coordinate systems. Both analytic and full geometry benchmark cases are presented.

1 INTRODUCTION

Fusion science in the 21st century is increasingly reliant on large scale computer simulations. This has led to an increasing need for verification and validation (V&V) [1]-[2] capability for large simulation codes.

Eigenvalue solvers for partial differential equations have a number of advantages as tools for verification of turbulence simulations. They are simpler than full nonlinear simulations, thus are inherently less prone to error and can be benchmarked adequately with a smaller and simpler set of test cases. They require significantly less computing resources, thus allowing tests to be done quickly and conveniently. Conversion of differential equations into matrix form can be separated from solution of eigenvalues, thus providing additional information for debugging purposes, as well as simplifying isolation of any errors that do arise.

The 2DX code is an eigenvalue solver designed to operate in an x-point topology. This makes it relevant to tokamak edge phyics, particularly modeling the interaction between edge, scrape-off layer, and divertor plasmas. As such, it can be used to benchmark turbulence codes simulating that region of the plasma. In this paper, we present benchmarking of 2DX with the BOUT [4]-[5] edge turbulence code.

In addition to its application as a benchmarking tool, the versatility of the 2DX code makes it very useful for physics applications. The ability to calculate the growth rates of eigenmodes quickly makes it useful in determining stability thresholds for plasma instabilities such as ELM's [6]-[7]. Also, knowing the

spatial structure of dominant eigenmodes can provide insight into turbulence, for instance allowing estimation of typical frequency and wavenumber bands for fluctuations. In the direction parallel to the background magnetic field in the scrape-off-layer plasma, the degree of connection of modes between the x-point region and divertor plates is of interest [8]-[9]. An application of 2DX to this problem is discussed in Sec. 4.2.3.

An additional noteworthy feature of the 2DX code is its use of a specialized input format for parsing systems of equations. This gives the code an exceptional degree of flexibility in handling different physics models. It also offers a number of advantages from the standpoint of code verification. First, it splits each problem into two parts: the equation language file, and the source code to parse that equation language. While this introduces a potential source of error each time a new set of equations is used, this problem can be easily isolated, thus accumulating confidence in the source code across many different benchmark cases. Second, the equation language file can be translated into analytic form, thus allowing the user to determine, in an intuitive manner, precisely what equations are being solved. This offers a considerable advantage over codes in which equation sets are hard-wired into the source code, which is difficult to read, mingles formula and numerical technique, and lacks concise expression. Moreover, it maximizes transparency in the area where errors are most likely to occur, and does so in a way that is accessible to the casual user.

2 PROGRAM STRUCTURE

The 2DX code consists of two main parts, as well as a number of tools required to set up its input files and process its output files. The relation between these parts is shown in Fig. 1. The tools shown in this figure (grid generation tools, structure file viewer, data analysis tools) are currently implemented as Mathematica worksheets, although Python scripts for grid generation are also available. The 2DX code handles input and output as text files. The input file contains lists of constants and input functions (including Jacobian factors) on the grid. For simple problems and geometries, the input file can be created rather easily by a small separate code or script. For complex problems, such as the divertor geometry problem to be considered in Sec. 4.2.1, we employ a Mathematica notebook that calculates functions such as coordinate systems curvatures, shear, and Jacobians in toroidal geometry. This notebook can perform high order interpolations to refine a magnetic geometry mesh that originates from experimental equilibrium reconstructions.

The 2DX output file contains the specified number of eigenvalues and corresponding eigenfunctions on the mesh. These are extracted and processed by separate data analysis tools. Again, for simple problems and geometries, a small plotting code is all that is required. For complex problems in toroidal geometry, Mathematica or IDL codes provide more sophisticated tools that make use of the magnetic topology. Examples will be discussed later, in connection with Figs. 9,10 and 16. Finally, the Structure file viewer, and the associated procedure for creating structure files, will be discussed in Sec 2.4.

The principal components of the 2DX code are the core 2DX program and the eigenvalue solver. The core 2DX code generates a pair of sparse matrices in coordinate list format (sometimes referred to as COO format), i.e. as a tuple containing the (value, row, column) of each nonzero entry. These matrices form a generalized eigenvalue problem,

$$Ax = \lambda Bx \tag{1}$$

The pair of matrices is passed to an eigenvalue solver, which returns the eigenvalues λ and eigenvectors x. In the current version of the code, the eigenvalue solver employed is SLEPc [3]. This eigensystem package can attain a solution in a number of different ways, depending on user-selected options. Experience to date has achieved best results using a combination of a Krylov-Schur algorithm [10] combined with a Cayley spectral shift technique [11]. The spectral shift re-organizes the eigenvalues so that the eigenvalues with the largest real part (i.e. the fastest growing modes, hence the modes of interest) are also the largest absolute value eigenvalues. This is important because in a typical eigenvalue problem arising from the solution of partial differential equations, the reverse will be true: the largest absolute value eigenmodes will tend to be high wavenumber, poorly resolved modes that are either strongly damped or are neutrally stable with high frequency. Since sparse eigenvalue solution techniques lose their advantage if more than a handful of eigenvalues are returned, and since full eigenvalue solution techniques are impractical for large problems due to unfavorable scaling, it is critically important to choose a solution method

that prioritizes the modes of interest correctly.

Most of the distinguishing features of 2DX are in the core 2DX program, and relate to how it sets up the eigenvalue problem. Matrices are built up from built-in finite difference operators, boundary condition operators, and diagonal matrices constructed from functions. These simple matrices are combined into more complicated matrices using matrix addition and multiplication operations. The sequence of these operations is controlled by a specialized equation parser, using input in a data format called the equation language. The equation language is capable of immense versatility; given a sufficiently large set of equation language instructions, virtually any finite difference method of finite order or virtually any boundary condition can be created from these basic building blocks. This type of construction makes the code exceptionally flexible in what types of problems it can solve or what numerical methods it can use. In addition, it means that the 2DX source code itself contains only instructions for creating elementary operator matrices, performing elementary matrix operations, and parsing equation language files. This results in a code that is short, simple, and easy to debug. The drawback to this approach is that it shifts much of the burden of debugging to the various input files and the tools used to create these. This drawback is addressed subsequently. The net benefit of this approach is that 2DX possesses a modular structure that cleanly separates these tasks.

2.1 Elementary operators

Matrices in 2DX are built up from elementary operators. In addition to diagonal matrices used to represent profile functions or other functions derived from them, there are four finite difference operators and up to six boundary operators. The elementary differential operators (denoted u and l for upper and lower) are:

$$(\partial_x^u)_{ix,iy;jx,jy} = \begin{cases} -\frac{1}{dx} & \text{if } ix = jx, iy = jy \\ \frac{1}{dx} & \text{if } ix + 1 = jx, iy = jy \end{cases} \tag{2}$$

$$(\partial_x^l)_{ix,iy;jx,jy} = \begin{cases} \frac{1}{dx} & \text{if } ix = jx, iy = jy \\ -\frac{1}{dx} & \text{if } ix - 1 = jx, iy = jy \end{cases} \tag{3}$$

$$(\partial_y^u)_{ix,iy;jx,jy} = \begin{cases} -\frac{1}{dy} & \text{if } ix = jx, iy = jy \\ \frac{1}{dy} & \text{if } ix = jx, iy + 1 = jy \end{cases} \tag{4}$$

$$(\partial_y^l)_{ix,iy;jx,jy} = \begin{cases} \frac{1}{dy} & \text{if } ix = jx, iy = jy \\ -\frac{1}{dy} & \text{if } ix = jx, iy - 1 = jy \end{cases} \tag{5}$$

The boundary operators are matrices that are zero everywhere except at diagonal entries corresponding to grid points on a boundary. Thus, there are boundary operators for the upper and lower boundaries in x and y. In addition, there are two additional boundary operators that are offset by one grid cell from the upper boundaries in x and y. These are used to create boundary conditions on staggered grids.

From these basic operator matrices, it is possible to build up differential operators of arbitrary order through successive addition and multiplication. For

instance, suppose one wants to create a central difference second derivative operator. In this case, one would simply multiply two elementary operators: $\partial_x^2 = \partial_x^u \partial_x^l$. Likewise, a central difference first derivative can be constructed by adding two elementary operators and dividing by two: $\partial_x^c = (\partial_x^u + \partial_x^l)/2$.

In practice, most models allow for non-uniform grids. To account for this, the elementary operators must be multiplied by appropriate profile functions containing geometry information. This results in operators more complicated than the ones described above, but the overall concepts involved remain the same.

2.2 Staggered grids

An option in the equation language file is to make certain variables indented. This means that the last row of grid cells in one or both directions is deleted for that variable. The purpose of this is to permit staggered grids. Since grid points for that variable have one less row in one direction, they can be thought of as being between grid points on a normal grid. By choosing appropriate differential operators (upper or lower) in that direction for all terms linking the indented variable to non-indented variables and vice versa, one can make this concept a numerical reality.

The primary purpose of staggered grids in this code is to deal with numerical issues arising when a second derivative of the eigenfunction arises from two first derivative operators applied to different fields. For instance, as we will see in Eq. 27-32, the $\delta\phi$ and δA equations interact via parallel derivatives, so that

in the high collisionality limit there is an effective second parallel derivative on $\delta\phi$. If two central difference operators are used for the first derivative, the resulting second derivative operator skips directly adjacent grid cells. As a result, spurious eigenmodes with large $k_\parallel$ emerge. Convolving first derivative operators that are offset in opposite directions yields the correct form for the second derivative operator, thus avoiding this problem. Given the underlying flexibility of the 2DX code, other applications of this capability (such as ensuring zero divergence of vector fields) are also possible.

2.3 Grid topology

In the present version of 2DX, the domain of the grid is divided into four regions. This feature can be easily generalized to handle more complicated topologies. The present form is suitable for a wide range of edge physics applications in tokamak divertor geometry.

One of these regions is the edge. This is the region inside the separatrix. It is bounded by periodic boundary conditions in y.

A second region is the scrape-off layer. This is the region outside the separatrix but adjacent to the edge. It is subject to sheath boundary conditions in y.

The other two regions are the private scrape-off layer. This is the region opposite the x-point from the edge. It is subject to both matching boundary conditions linking the two pieces of the private region, as well as sheath boundary conditions. In allocating space on the grid, it is located on either side of the

edge in y, and adjacent to the SOL in x.

The layout of these regions is shown in Fig. 2. Note that the x-point lies in the interstices between grid points, thus the singularity at the x-point is avoided. Because of the periodic boundary condition in the edge region, all eight grid points adjacent to the x-point connect to each other to form an octagonal cell, in contrast to the quadrilateral cells formed by other cycles of adjacent points.

2.4 Equation language

The equation langauge is a data file format containing all of the information the 2DX code needs to convert profile functions (i.e. temperature, density, magnetic geometry) into matrices to solve. This consists of a number of parts, of which three are of particular importance. These are the input language, the element language, and the formula language.

The input language defines the format of data files containing integer and real constants and profile functions. Here, the term profile function refers to an input quantity (for example a coefficient of the differential equations) that is specified on the 2D grid. Each entry in the input language consists of a data label and instructions on what to do with the actual data. This allows the 2DX code to determine whether a data block is supposed to contain an integer, a real number, or a profile function. Moreover, each data block is assigned a unique identifier so that it can be referenced by other parts of the equation language.

The element language consists of a series of basic operations that can be applied to profile functions or to operators. This is used to build up all of

the differential operators used in an equation set, as well as any derived profile functions (i.e. functions calculated from other functions). This is done by successive application of basic binary or unary algebraic or matrix operations. For algebraic operations, the current version of 2DX permits addition, subtraction, multiplication, division, powers, exponentials, and linear interpolation. For matrix operations, it permits addition, subtraction, multiplication, inverse, and transpose.

The formula language is used to specify the eigenvalue equations from the basic operators and functions created by the element language. Thus it generates the matrices required for a generalized eigenvalue problem. Since the equations of interest may contain more than one field, the eigenvector and matrices are larger than the number of grid points. That therefore means that the matrices will be larger than the elementary functions and differential operators created by the other parts of the language. To accomodate this, the 2DX code first multiplies together a string of functions and operators to form a matrix block. This block is then offset by adding integer multiples of its own size to the row and column indices of the elements of that block; this is straightforward to do for a sparse matrix in coordinate form, since row and column indices are simply integers associated with each nonzero entry. Adding to row indices determines which equation the term is in, whereas adding to the column index determines which field the term multiplies. By adding together a succession of such terms, the 2DX code is able to construct matrices to represent nearly arbitrary sets of equations provided they correspond to a linear eigenvalue problem.

The equation language can be converted via Python scripts between a numerical format that can be read by 2DX and a symbolic format that can be read by the user. In addition, a Mathematica script can be used to translate the numerical format into a standard algebraic form using symbolic logic.

For a simple example of how the equation language works, consider the following eigenvalue problem for $\delta\Phi$:

$$\frac{\partial}{\partial t}\nabla_\perp^2\delta\Phi = \mu_{ii}\nabla_\perp^4\delta\Phi \tag{6}$$

This can be coded using the symbolic format of the equation language as:

$$gg * dprp2 * PHI = muii * dprp4 * PHI \tag{7}$$

where the element language contains instructions for building the operators dprp2 and dprp4 from elementary operators. This can then be translated into the numerical format and read by the Mathematica script to yield the following:

$$\nabla_\perp^2\lambda\delta\Phi = muii\nabla_\perp^4\delta\Phi \tag{8}$$

where λ is the eigenvalue. In this particular case, the element language is only used to define operators. In cases where the element language is used to define functions, the equation viewer will unravel these equations so as to display algebraic expressions of profile functions whenever possible.

The combination of symbolic format structure files and the Mathematica viewer script provides a layer of protection against coding errors in the equation language. Since the viewer script uses the same parsing logic as the 2DX code, it therefore displays the equations as they will actually be run. Thus, if a term

in the structure file is coded incorrectly, a discrepancy between the input and viewer output will become evident.

3 PHYSICAL MODEL

3.1 Coordinate geometry and differential operators for tokamak edge plasma

The equation language files used in the test cases presented later in this article are based on a ballooning (field line following) coordinate system. This coordinate system is also used by BOUT[4]-[5]. This coordinate system is defined by:

$$x = \psi - \psi_s \tag{9}$$

$$y = \theta \tag{10}$$

$$z = \zeta - \int_{\theta_0} d\theta \nu(\psi, \theta) \tag{11}$$

where ζ is the toroidal angle, θ is the poloidal angle, ψ is poloidal flux, and ν is the local safety factor.

Using toroidal symmetry of the equilibrium, we assume that the solution is periodic in ζ. Then, by specifying the eigenfunction in the form:

$$\delta\phi = \phi_1(x, y)e^{inz} \tag{12}$$

the potentially rapid (for large n) phase variation is extracted and the numerics need only resolve the coefficient ϕ_1. Although closely related to the eikonal approximation [23] (a local approximation valid for $n \gg 1$, where ϕ_1 can be taken as a function of y alone) , Eq. 12 and the numerical formulation can in principle be made exact (for infinite grid resolution).

The magnetic field can be calculated from the coordinates by introducing a local "safety factor" ν. This results in the following definition of B:

$$\mathbf{B} = \nu \nabla\psi \times \nabla\theta + \nabla\zeta \times \nabla\psi \tag{13}$$

From this field-line following coordinate system we derive formulas for parallel and perpendicular gradients based on coordinate derivatives. For this purpose we must construct profile functions containing relevant information about the structure of the magnetic field. The principal geometry profile functions used are:

$$\jmath = \nabla\psi \times \nabla\theta \cdot \nabla\zeta / B \tag{14}$$

$$RB_p = |\nabla\psi| \tag{15}$$

$$k_b = -nB/RB_\theta \tag{16}$$

$$k_\psi = -nRB_p \left(\frac{\nu \nabla\theta \cdot \nabla\psi}{RB_p^2} + \int_{\theta_0} \frac{\partial}{\partial\psi} \nu \right) \tag{17}$$

$$\kappa_g = \kappa \cdot \hat{b} \times \hat{e}_\psi \tag{18}$$

$$\kappa_n = \kappa \cdot \hat{e}_\psi \tag{19}$$

where

$$\kappa = \hat{b} \cdot \nabla \hat{b} \tag{20}$$

$$\hat{e}_\psi = \frac{\nabla \psi}{|\nabla \psi|} \tag{21}$$

and B_θ is the poloidal field. The Jacobian quantities $\jmath$ and RB_p define the physical distances corresponding to the grid spacing, the quantities k_b and k_ψ define the eikonal wavenumbers orthogonal to the magnetic field in the $\hat{b} \times \hat{e}_\psi$ and $\hat{e}_\psi$ directions [23], and the quantities κ_g and κ_n are components of field-line curvature.

Making the simplifying but non-essential assumption that parallel derivatives are small compared to perpendicular derivatives, we then derive the following basic operators:

$$\nabla_\parallel = \jmath \frac{\partial}{\partial y} \tag{22}$$

$$\nabla_\perp^2 = -k_b^2 - B\jmath \left(k_\psi - i\partial_x RB_p\right) \frac{1}{B\jmath} \left(k_\psi - iRB_p \partial_x\right) \tag{23}$$

$$C_r = -\kappa_g RB_p \partial_x + i\kappa_n k_b - i\kappa_g k_\psi \tag{24}$$

Of these operators, the curvature operator C_r is constructed by averaging the operators ∂_x^u and ∂_x^l to give a central difference first derivative operator with second order accuracy. The operator $\nabla_\perp^2$ alternates the use of ∂_x^u and ∂_x^l so as to yield a centered second derivative operator, also with second order accuracy. The operator $\nabla_\parallel$ is used in terms that link variables on the staggered grid to variables not on the staggered grid; for this reason, a single operator of the type ∂_y^u or ∂_y^l is sufficient to achieve a central difference first derivative

operator. Since operators of opposite types are used when going to and from the staggered grid, successive operations of this type alternate upper and lower derivatives so as to yield a centered second derivative operator.

3.2 Boundary conditions

Boundary conditions in the x direction are built into operators involving x derivatives, by adding suitable multiples of the boundary operators. For all of the test cases listed in the following section, the x boundary employs a zero derivative boundary condition. This is chosen in order to permit the 2D system to allow a 1D local limit, i.e. ϕ_1 constant in x, thus facilitating comparison to local analytic theory.

In the y direction, boundary conditions are more complicated. First of all, there are two types of boundary conditions: phase-shift periodic and sheath, with matching boundary conditions treated as a special case of phase-shift periodic. The layout of these is described in Fig. 2.

In the case of phase-shift periodic boundary conditions, the goal is to modify the differential operators so as to apply smoothly to the solution in field-line following coordinates, as described by the ϕ_1 term in Eq. 12. Since the periodic boundary condition also serves as a branch cut in z, it follows that a phase shift must be applied at the branch cut. Noting that:

$$\phi_1(y = 0) = \phi_1(y = 2\pi)e^{-2\pi i n q} \tag{25}$$

and also noting that the normal off-diagonal terms of the ∂_y operators are equal to $1/dy$, it follows that the off-diagonal terms that need to be added

to the matrix to achieve the phase-shift periodic boundary conditions can be constructed by applying conjugation or sign inversion to terms of the form:

$$\frac{e^{2\pi i n q}}{dy} \tag{26}$$

Sheath boundary conditions depend on the underlying physics model. Because they can couple multiple fields, they are implemented as separate terms in the model equations rather than as modifications to existing operators. For instance, the boundary condition in a parallel current equation will typically depend on potential, temperature, and density. In these cases an appropriate formula is multiplied by a sum or difference of the built-in boundary condition operators. This approach permits the flexbility needed to simulate diverse physical boundary conditions.

3.3 6-field fluid model for collisional plasma

While the 2DX code is capable of solving eigenvalues for a wide variety of equation systems, actually exercising this capability represents a potential source of error. Each time a new equation set is converted into equation language form, there is a possibility that one of the instructions in that file is incorrect. To minimize this source of error, most of the test cases presented use subsets of a standardized physics model. The standardized physics model is a linearized version of BOUT equations [4]-[5], thus simplifying comparison between the two codes. The model equations are expressed in dimensionless Bohm units, where time is normalized to the ion cyclotron frequency Ω_i and length is normalized to the cold ion sound gyroradius $\rho_s = c_s/\Omega_i$ with $c_s^2 = T_e/m_i$. The full set of

model equations considered here is as follows:

$$\gamma\nabla_\perp^2\delta\Phi = -i\omega_{*i}\nabla_\perp^2\delta\Phi + \frac{2B}{n}C_r\delta p - \frac{B^2}{n}\partial_\parallel\nabla_\perp^2\delta A + \Gamma\nabla_\perp^2\delta\Phi + \mu_{ii}\nabla_\perp^4\delta\Phi \quad (27)$$

$$\gamma\delta n = -\delta v_E\cdot\nabla n + \frac{2}{B}\left(C_r\delta p_e - nC_r\delta\Phi\right) - n\partial_\parallel\delta u - \partial_\parallel\nabla_\perp^2\delta A \quad (28)$$

$$\gamma\delta u = -\frac{1}{n}\nabla_\parallel\delta p - \frac{1}{n}\delta b\cdot\nabla p - \frac{2T_i}{B}C_r\delta u + \partial_\parallel\mu_\parallel\nabla_\parallel\delta u \quad (29)$$

$$\gamma\delta T_e = -\delta v_E\cdot\nabla T_e - \frac{2(1.71)T_e}{3n}\partial_\parallel\nabla_\perp^2\delta A - \frac{2}{3}T_e\partial_\parallel\delta u$$
$$+\frac{2}{3}\partial_\parallel\chi_\parallel\left(\nabla_\parallel\delta T_e + \delta b\cdot\nabla T_e\right) + \frac{4T_e}{3B}\left(\frac{1}{n}C_r\delta p_e - C_r\delta\Phi + \frac{5}{2}C_r\delta T_e\right) \quad (30)$$

$$\gamma\delta T_i = -\delta v_E\cdot\nabla T_i - \frac{2T_i}{3n}\partial_\parallel\nabla_\perp^2\delta A - \frac{2}{3}T_i\partial_\parallel\delta u$$
$$+\frac{4T_i}{3B}\left(\frac{1}{n}C_r\delta p_e - C_r\delta\Phi - \frac{5}{2}C_r\delta T_i\right) \quad (31)$$

$$\gamma\left(\frac{n}{\delta_{er}^2} - \nabla_\perp^2\right)\delta A = \nu_e\nabla_\perp^2\delta A - \mu n\nabla_\parallel\delta\Phi + \mu T_e\nabla_\parallel\delta n + \mu T_e\delta b\cdot\nabla n$$
$$+1.71\mu n\nabla_\parallel\delta T_e + 1.71n\mu\delta b\cdot\nabla T_e \quad (32)$$

where γ is the eigenvalue with real part corresponding to growth rate, $\mu = m_i/m_e$, and δ_{er} is the reference electron skin depth. Other notations are standard (see e.g. Refs. [4],[5]). In addition, we define the following quantities:

$$\omega_{*i} = \frac{k_b\partial_r p_i}{nB} \quad (33)$$

$$\delta p = (T_e + T_i)\delta n + n(\delta T_e + \delta T_i) \quad (34)$$

$$C_r = \mathbf{b}\times\kappa\cdot\nabla \quad (35)$$

$$\partial_\parallel Q = B\nabla_\parallel(Q/B) \quad (36)$$

$$\delta v_E\cdot\nabla Q = -i\frac{k_b(\partial_r Q)}{B}\delta\Phi \quad (37)$$

$$\delta\mathbf{b}\cdot\nabla Q = i\frac{k_b(\partial_r Q)}{\delta_{er}^2 B\mu}\delta A \quad (38)$$

The sheath boundary conditions for this model are as follows:

$$-\nabla_\perp^2 \delta A = -\sigma n (T_e + T_i)^{1/2} \left(\frac{1}{2} \frac{\delta T_e + \delta T_i}{T_e + T_i} - \frac{1}{2} \frac{\delta T_e}{T_e} + \frac{\delta \Phi}{T_e} \right) \tag{39}$$

$$\delta u = -\sigma \frac{\delta T_e + \delta T_i}{2(T_e + T_i)^{1/2}} \tag{40}$$

$$\delta T_e = \sigma \frac{4}{9} \frac{\chi_\parallel}{S_E T_e^{1/2}} \left(\nabla_\parallel \delta T_e + \delta \mathbf{b} \cdot \nabla T_e \right) \tag{41}$$

where $\sigma = \pm 1$.

Of the fields presented in this model, δu and δA are indented in y, i.e. they use a staggered grid in the y direction. Thus, whereas the operator $\partial_\parallel$ used in Eqs. 27,28,30, and 31 uses the lower derivative ∂_y^l defined in Eq. 5, the operator $\nabla_\parallel$ used in Eqs. 29 and 32 uses the upper derivative ∂_y^u defined in Eq. 4. This is in addition to the indented fields being evaluated on one less row of grid cells in y than the other fields.

4 VERIFICATION AND TEST CASES

The 2DX code has been benchmarked against a number of different test cases. Initial tests (not shown here) successfully reproduced analytical solutions of the 2D quantum harmonic oscillator problem and served to verify the basic 2DX solver kernel. A suite of additional tests was developed to further test 2DX together with the structure file associated with the six-field model. These are described in detail in the present section.

Tokamak edge plasma combines complex magnetic geometry and very rich physics, even in the framework of a linearized fluid model. The full two-fluid

six-field model has numerous instability branches, which makes it challenging to verify the full model. A verification approach taken here is testing several selected instabilities standalone by reducing the full system to a particular subset of equation terms that support a selected branch. Most of these cases permit comparison to BOUT results. In addition, a number of them have analytic solutions that results from both 2DX and BOUT can be compared to.

The analytic test cases were chosen in order to provide a representative sample of instabilities commonly encountered in tokamak edge turbulence. The resistive ballooning mode is important in the edge and scrape-off layer, and the resistive drift wave is important in the plasma edge, particularly for steep profiles. The electromagnetic version of the drift wave also tests Alfven wave physics, known to be important for edge plasmas. Ion temperature gradient modes are important throughout the plasma when the temperature profile varies more rapidly than the density profile. In addition, the geodesic acoustic mode was chosen in order to demonstrate the ability of the code and its equation set to model more complex observed physical phenomena that depend in an essential way on the toroidal geometry.

Thus the sequence of sub-models tests a large variety of physically relevant terms from the full six-field model. Importantly the tests also progress from simple slab geometries through more complex idealized tokamak models, and finally to full x-point divertor geometry. For full geometry tests, test models were chosen based on known instabilities relevant in their respective regions of interest. In addition, simple models were chosen in order to isolate any numerical

issues relating to the more complex topology or non-uniform grid these cases present.

4.1 Analytic tests

4.1.1 Resistive ballooning model

The resistive ballooning model [12]-[14] uses the following subset of the 6-field model:

$$\gamma \nabla_\perp^2 \delta\Phi = +\frac{2B}{n} C_r \delta p - \frac{B^2}{n} \partial_\parallel \nabla_\perp^2 \delta A \tag{42}$$

$$\gamma \delta n = -\delta v_E \cdot \nabla n \tag{43}$$

$$-\gamma \nabla_\perp^2 \delta A = \nu_e \nabla_\perp^2 \delta A - \mu n \nabla_\parallel \delta\Phi \tag{44}$$

For the analytic test case, the profile functions are set so as to create a homogenous problem; this simplifies calculation of analytic solutions. Specifically, the analytic geometry is a twisted annulus with a q of 1.5, and phase shift periodic boundary conditions in the parallel direction. Density has an exponential profile, and temperature is constant. Toroidal mode number is then calculated from the variable k_b by the formula $n = k_b a/q$ where a is the radius of the annulus.

The results from this test are compared to analytic approximations in the high and low k_z cases. In addition, this model has also been simulated using BOUT. This provides further validation of the 2DX code.

The results from this are shown in Fig. 3. Circles represent eigenvalues

calculated using 2DX. The curved lines are analytic approximations, and the crosses are growth rates calculated from BOUT with associated error bars. Since BOUT is a simulation and not an eigenvalue solver, the growth rates shown are calculated by fitting an exponential to fluctuation amplitude, and deviation from a pure exponential (for instance due to mode contamination) is used to calculate the error bars.

4.1.2 Resistive drift wave model

Two resistive drift wave models [15]-[16] were tested, one retaining only electrostatic terms, one including electromagnetic terms. The electrostatic model is as follows:

$$\gamma \nabla_\perp^2 \delta\Phi = -\frac{B^2}{n} \partial_\parallel \nabla_\perp^2 \delta A \tag{45}$$

$$\gamma \delta n = -\delta v_E \cdot \nabla n \tag{46}$$

$$-\gamma \nabla_\perp^2 \delta A = \nu_e \nabla_\perp^2 \delta A - \mu n \nabla_\parallel \delta\Phi + \mu T_e \nabla_\parallel \delta n \tag{47}$$

The electromagnetic model is as follows:

$$\gamma \nabla_\perp^2 \delta\Phi = -\frac{B^2}{n} \partial_\parallel \nabla_\perp^2 \delta A \tag{48}$$

$$\gamma \delta n = -\delta v_E \cdot \nabla n \tag{49}$$

$$\gamma \left(\frac{n}{\delta_{er}^2} - \nabla_\perp^2 \right) \delta A = \nu_e \nabla_\perp^2 \delta A - \mu n \nabla_\parallel \delta\Phi + \mu T_e \nabla_\parallel \delta n + \mu T_e \delta b \cdot \nabla n \tag{50}$$

Both of these models were tested in a slab geometry with periodic boundary conditions. Unlike the resistive ballooning model, the test case required some

special features to deal with properties of the model equations. In particular, these model equations yield the maximum growth rate when both k_x and k_y are large and in a particular proportion to one another; if growth rate is plotted as a function of these parameters, it displays a long ridge with height asymptotically approaching a maximum value. As a result, if high resolution is available in both the x and y directions, the model will produce a dominant eigenmode that is not well resolved in at least one of those directions. To solve this problem, resolution in the x direction is highly restricted, and the grid size is made very large so that the maximum resolvable derivative in that direction is small; the latter restriction is necessary because the 2DX code cannot handle resolution less than two in either direction. The resulting test case closely resembles a 1D model, and can therefore be easily compared to analytic theory. Moreover, it results in a dominant eigenmode that is well-resolved.

In this geometry, the model is compared to analytic theory and BOUT results for a number of values of $k_\parallel$, which is controlled by adjusting the size of the domain. In order to compare the model results without regards for dimensional quantities, the modes are de-dimensionalized by constructing variables as follows:

$$\omega_* = k_\perp v_{pe} \equiv k_\perp \frac{v_{te}^2}{\omega_{ce} L_n} \tag{51}$$

$$\sigma_\parallel = \left(\frac{k_\parallel}{k_\perp}\right)^2 \frac{\Omega_{ci}\omega_{ce}}{0.51\nu_{ei}} \tag{52}$$

$$\sigma_\perp = 0.51\nu_{ei}\mu \tag{53}$$

The results from this comparison are shown in Fig. 4 - 5.

For domains with low minimum $k_\parallel$, the dominant eigenmode will typically have a higher $k_\parallel$ than the minimum allowed. This results in a spurious eigenmode that masks actual trends in growth rate as a function of $k_\parallel$. To solve this problem, the code is set to return a number of eigenvalues. Their associated eigenfunctions are then sorted by $k_\parallel$ as measured by comparing phase angles of adjacent grid points. This permits the lowest $k_\parallel$ eigenvalue to be extracted and compared to analytic theory.

4.1.3 Slab ion temperature gradient mode model

The slab ion temperature gradient mode model [17]-[18] uses the following subset of the 6-field model:

$$\gamma \nabla_\perp^2 \delta\Phi = -\frac{B^2}{n} \partial_\parallel \nabla_\perp^2 \delta A \tag{54}$$

$$\gamma \delta n = -n \partial_\parallel \delta u - \partial_\parallel \nabla_\perp^2 \delta A \tag{55}$$

$$\gamma \delta u = -\frac{1}{n} \nabla_\parallel \delta p \tag{56}$$

$$\gamma \delta T_i = -\delta v_E \cdot \nabla T_i - \frac{2}{3} T_i \partial_\parallel \delta u \tag{57}$$

$$-\gamma \nabla_\perp^2 \delta A = -\mu n \nabla_\parallel \delta\Phi + \mu T_e \nabla_\parallel \delta n \tag{58}$$

As with the resistive drift wave model, the ITG model is tested in a geometry with limited resolution in the x direction, in order to solve the equations in an effectively 1D limit. Unlike the resistive drift wave case, with the ITG test the code is set to return multiple eigenvalues, from which a parallel wavenumber

can be calculated from the spatial structure of the eigenmode. This allows an entire mode spectrum to be derived from a single run. More importantly, it returns some modes from the neutrally stable branch of the ITG solution.

The results from this test are compared to analytic theory as well as to runs from BOUT. The results from this are shown in Fig. 6.

4.1.4 Geodesic acoustic mode model

The geodesic acoustic mode model [19] uses the following subset of the 6-field model:

$$\gamma\nabla_\perp^2\delta\Phi = +\frac{2B}{n}C_r\delta p - \frac{B^2}{n}\partial_\parallel\nabla_\perp^2\delta A + \Gamma\nabla_\perp^2\delta\Phi + \mu_{ii}\nabla_\perp^4\delta\Phi \tag{59}$$

$$\gamma\delta n = +\frac{2}{B}\left(C_r\delta p_e - nC_r\delta\Phi\right) - n\partial_\parallel\delta u - \partial_\parallel\nabla_\perp^2\delta A \tag{60}$$

$$\gamma\delta u = -\frac{1}{n}\nabla_\parallel\delta p \tag{61}$$

$$-\gamma\nabla_\perp^2\delta A = \nu_e\nabla_\perp^2\delta A - \mu n\nabla_\parallel\delta\Phi + \mu T_e\nabla_\parallel\delta n \tag{62}$$

The parameter Γ is set to a positive value in this model in order to provide an instability drive for the GAM. Physically, this represents coupling of the GAM to turbulence.

The geometry used in the GAM test is more complicated than in the previous test cases. This is because the GAM can only exist if there is geodesic curvature. Instead of a simple slab model, the GAM test is performed in an idealized torus. Also, instead of performing a scan in parallel wavenumber, a scan is performed instead in the q of the idealized torus, i.e. one with circular flux surfaces and

low inverse aspect ratio.

The results from this are shown in Fig. 7. This plot compares the results from the 2DX code to two different analytic solutions: one for the theoretical GAM neglecting mode coupling (see below), and one for a zonal flow. Since both modes are solutions of the model equation, and since the 2DX code returns the dominant eigenvalue, both solutions are needed for comparison. The zonal flow solution, which dominates at low q, matches the 2DX result in that regime, whereas the analytic GAM matches the 2DX result in the high q regime.

A peculiar anomaly in this test is the presence of regularly spaced deviations between the 2DX result and the analytic GAM. This is due to coupling with sound waves. Taking into account coupling between the GAM at frequency:

$$\omega_g^2 = \frac{1}{q^2 R^2} + \frac{2}{R^2} \tag{63}$$

and spatial harmonics of the sound wave at frequency:

$$\omega_s^2 = \frac{m^2}{q^2 R^2} \tag{64}$$

predicts mode coupling, and hence modification of the simple result, when:

$$q = \sqrt{\frac{m^2 - 1}{2}} \tag{65}$$

These results agree with the position of the deviations in Fig. 7.

4.2 Full geometry tests

4.2.1 DIII-D test cases

The 2DX code has been tested and cross-benchmarked against BOUT in a full geometry case based on the edge plasma of D-IIID [26]. The model tested was the resistive ballooning model. The pressure profile was modified somewhat in order to avoid having a dominant mode localized along the radial boundary, but the magnetic geometry is fully realistic.

To best verify the capabilities of the 2DX code, we sought a scalar parameter scan in which the eigenmodes vary from broad (to sample the magnetic geometry) to localized (to permit comparison with an local analytical theory). Such a test can be achieved by varying collisionality, through Z_{eff}. For relatively low values of collisionality, such as occur for these profiles with $Z_{eff} = 1$, and a torodial mode number n = 100, the modes fill the torus (but are stronger on the outboard side). Increasing Zeff to artificially large values for purposes of the verification test, we find eventual saturation of the growth rate with Z_{eff} as shown in Fig.8 as the modes collapse down to a local point in poloidal angle on the low field side. This full range of conditions is adequately benchmarked in this test between 2DX and BOUT. To obtain agreement for the low gamma, Z_{eff}= 1 case, it was necessary to employ finite μ_{ii}, so that grid-induced dissipation in BOUT did not influence gamma (a rather low resolution was employed for these tests in BOUT).

Also shown in the figure are the 2DX results for $n = 10^3$ and 10^4. As n

is increased the eigenfunctions also collapse in radial location, and the growth rate asympotically converges to $4.83 \times 10^5/s$.

A simple local analytical result for the resistive ballooning mode is possible in the limit where Z_{eff} and n are both asymptotically large:

$$\gamma = c_s \sqrt{\frac{2}{R_{eff} L_n}} \tag{66}$$

This result, shown by a green dashed line in Fig. 8 is also at $4.83 \times 10^5/s$. Thus by a sequence of steps we have connected the BOUT and 2DX codes for realistic tokamak parameters to a solid analytical asymptotic result.

In addition to growth rates, eigenmode structures can also be compared. Figs. 9-10 show a comparison of eigenmode structures for this model for Z_{eff}=1. Fig. 9 is the 2DX result, while Fig. 10 is the BOUT result. While the two results are not entirely identical, this can be attributed to differences in grids, as well as differences in how the two codes handle branch cuts. In the latter case, such an issue can arise because 2DX applies a phase shift at the branch cut in Fourier space as described by Eq. 25, whereas BOUT applies this phase shift in real space as described in Ref. [4].

Finally, the n=100 $Z_{eff} = 1$ case, being representative of a realistic tokamak problem of research interest, was used to analyze the scaling of the computational cost and numerical accuracy of the 2DX code. A number of test cases were generated by interpolating the original profiles at varying resolution, and the run time and leading eigenmodes were compared. Relative error was calculated by assuming the correct value to be an asymptote to a power law fit. In

addition, the time required to set up the matrix was compared to the time spent by SLEPc to solve it. These results are shown in Fig. 11-12. We conclude that 2DX displays near-linear convergence with grid size for this problem, and that run time is dominated by the SLEPc eigenvalue solution. It should be noted that the SLEPc package is parallelized, but the present test was run in serial mode on a single Intel-PC processor.

4.2.2 LAPD test cases

A number of test cases were done in order to verify the potential of the 2DX code for physics applications relevant to the Large Plasma Device (LAPD) [25]. Since LAPD is an open field-line device with a straight magnetic field, generating the correct geometry for this is simply a matter of using the SOL region of the 2DX grid and applying geometric profile functions to create a cylindrical coordinate system. In this case the x direction corresponds to radius (or more accurately enclosed flux), and the y direction corresponds to the z direction in standard cylindrical coordinates. Azimuthal angle is handled by toroidal mode number.

On this grid, the electrostatic resistive drift wave model (see Sec. 4.1.2) was simulated using 2DX and compared to a 1D eigenvalue solver. Temperature profile was assumed to be flat, and density profile was assumed to be of the form:

$$n(r) = a_0 + a_1 \frac{(1 + a_4 x)e^x - e^{-x}}{e^x + e^{-x}} \tag{67}$$

$$x = \frac{a_2 - r}{a_3} \tag{68}$$

The results of this are shown in Figs. 13-14. From this it can be seen that the 2DX results and the 1D eigenvalue results are in reasonable agreement.

4.2.3 Connection length study

In the edge region of a tokamak plasma, resistive ballooning instabilities can give rise to coherent turbulent structures which are localized perpendicular to **B** and extended along **B**. These structure are often referred to as blobs or blob-filaments [20]. These structures ultimately carry energy to the divertor plates or walls of the device; hence, it is of interest to understand the parallel structure and length scale (i.e. connection length) of the filaments. Experiments have attacked this problem by looking for correlations between the fluctuations at the midplane, X-point and divertor regions [8]-[9]. Midplane to X-point correlations have been seen [8] while an examination of midplane to divertor correlations [9] suggests a connection only when the fluctuations are sufficiently far radially from the X-point, in agreement with some earlier analytical theory [21] and eikonal and numerical studies [23]-[24]. We show here that 2DX can shed insight on this problem. It is one example of a physics application of the 2DX code which utilizes the full divertor geometry capability.

We began with experimentally measured profiles and geometry for a discharge on the National Spherical Torus Experiment [22]. To model a blob filament, we introduced a local bump in the radial pressure profile. The radial location of this bump was varied as shown in Fig. 15 with case a) on the closed surfaces, case b) on the separatrix, and case c) entirely in the open field line

region. The resulting eigenmode structures for each case are shown in Fig. 16. These show an eigenmode that is stopped in the parallel direction at the X-point when the bump is on the separatrix, but is more extended in the parallel direction when the mode is radially further away from the x-point. Note in particular that in case b) the mode does not reach the divertor plate, but in case c) it does, as found in Ref. [9].

5 SUMMARY

A new eigenvalue solver, the 2DX code, has been developed. It is capable of solving 2D linear partial differential equations in an x-point, periodic, or open field line topology. While designed specifically for problems in plasma physics pertaining to the edge of tokamaks, it is an immensely flexible code capable of solving a wide variety of problems.

The 2DX code has been tested against a number of cases, both in simple analytic geometry and in magnetic geometry derived from plasma experiments. These tests have been compared to analytic expressions and simulations of BOUT. Both comparisons have produced positive results.

The 2DX code shows great potential both as a benchmarking tool for plasma turbulence simulations and for direct physics applications. In the former case, it provides a simple code to which more complex codes can be compared. In the latter case, the ability to determine the spatial structure and growth rates of dominant eigenmodes of a system without the computational cost of a full

simulation (typically tens of CPU-hours for BOUT compared to a few CPU minutes for 2DX) can provide useful insight even in turbulent systems where dominant eigenvalues alone do not fully characterize its behavior.

Acknowledgements

This work was supported by the U.S. Department of Energy under grant DE-FG02-07ER84718 and by LLNL under DOE contract DE-AC52-07NA27344.

References

[1] X.Q. Xu and R.H. Cohen, Contrib. Plasma Phys. **36**, 158 (1998).

[2] M.V. Umansky et al., Contrib. Plasma Phys. **44**, 182 (2004).

[3] www.grycap.upv.es/slepc/

[4] M.V. Umansky, X.Q. Xu, B. Dudson, L.L. LoDestro, J.R. Myra, Computer Phys. Comm. **180**, 887 (2009)

[5] X.Q. Xu, et al., Phys. Plasmas **7** 1951, (2000).

[6] P. B. Snyder, et al., Nucl. Fusion **47**, 961 (2007).

[7] B. J. Burke, S. E. Kruger, C. C. Hegna, P. Zhu, P. B. Snyder, C. R. Sovinec and E. C. Howell, Phys. Plasmas **17**, 032103 (2010).

[8] J. L. Terry, S. J. Zweben, M. V. Umansky, et al., J. Nucl. Mater. **390**, 339 (2009).

[9] R. J. Maqueda, R. Maingi and NSTX team, Phys. Plasmas **16**, 056117 (2009).

[10] V. Hernandez, J. E. Roman, A. Tomas, V. Vidal, SLEPc technical report 7.

[11] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst (eds.) (2000). Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide. Society for Industrial and Applied Mathematics, Philadelphia, PA.

[12] H. Strauss, Phys. Fluids **24**, 2004 (1981).

[13] T. C. Hender, B. A. Carreras, W. A. Cooper, J. A. Holmes, P. H. Diamond, and P. L. Similon, Phys. Fluids **27**, 1439 (1984).

[14] P. N. Guzdar and J. F. Drake, Phys. Fluids B **5**, 3712 (1993).

[15] W. Horton, Rev. Mod. Phys. **71**, 735 (1999).

[16] M. Wakatani and A. Hasegawa, Phys. Fluids **27**, 611 (1984).

[17] B. Coppi, M.N. Rosenbluth and R.Z. Sagdeev, Phys. Fluids **10**, 582 (1967).

[18] S. Hamaguchi and W. Horton, Phys. Fluids B **2**, 1833 (1990).

[19] N. Winsor, J.L. Johnson and J.M. Dawson, Phys. Fluids **11**, 2448 (1968).

[20] S. I. Krasheninnikov, D. A. D'Ippolito, and J. R. Myra, J. Plasma Physics **74**, 679 (2008).

[21] R. H. Cohen and D. D. Ryutov, Contrib. Plasma Phys. **46**, 678 (2006).

[22] M. Ono, et al., Nucl. Fusion **40**, 557 (2000).

[23] J. R. Myra, D.A. DIppolito, X.Q. Xu and R.H. Cohen, Phys. Plasmas **7**, 2290 (2000).

[24] X.Q. Xu, R.H. Cohen, W.M. Nevins, G.D. Porter, M.E. Rensink, T.D. Rognlien, J. R. Myra, D.A. DIppolito, R.A. Moyer, P.B. Snyder and T.N. Carlstrom, Nucl. Fusion **42**, 21 (2002).

[25] W. Gekelman, H. Pfister, Z. Lucky, J. Bamber, D. Leneman, and J. Maggs, Rev. Sci. Instrum. **62**, 2875 (1991).

[26] J. L. Luxon, Nucl. Fusion **42**, 614 (2002).

Figure captions

Figure 1: Flowchart of data flow through the 2DX package. Modularization of the code into source code proper (2DX and SLEPc) as distinct from geometry (grid file) or equation set (structure file) has enabled rapid development and verification of the code.

Figure 2: Layout of the 2DX grid topology in the case of a single-null diverted tokamak. Cells that are adjacent on this chart are treated as adjacent by the code with the exception of cells adjoining the blue or purple lines; these are subject to a wraparound condition making them adjacent to more distant cells as indicated by the arrows.

Figure 3: Growth rate vs. k_b for resistive ballooning model. Black dots represent 2DX results, blue crosses represent BOUT results. The red curves are analytic approximations for large or small k_b, whereas the dashed line is an asymptotic solution for large k_b.

Figure 4 Growth rate vs. $\sigma_\parallel$ for resistive drift wave. The blue and purple lines are analytic solutions for the electrostatic and electromagnetic models, respectively. The blue and green dots are the fastest growing eigenvalues from a 2DX run, whereas the orange and red dots (obscured by the blue and green dots for large $\sigma_\parallel$) are the eigenvalues corresponding to the longest wavelength eigenmodes from the same run.

Figure 5 Frequency vs. $\sigma_{\parallel}$ for resistive drift wave. The blue and purple lines are analytic solutions for the electrostatic and electromagnetic models, respectively. The blue and green dots are the fastest growing eigenvalues from a 2DX run, whereas the orange and red dots (obscured by the blue and green dots for large $\sigma_{\parallel}$) are the eigenvalues corresponding to the longest wavelength eigenmodes from the same run.

Figure 6: Growth rate vs. $k_{\parallel}$ for ion temperature gradient mode model. Blue crosses are eigenvalues from a 2DX run as functions of the $k_{\parallel}$ value calculated from each corresponding eigenmode. Red crosses are analytic solutions calculated at the same $k_{\parallel}$ values as 2DX eigenmodes. The green circles are BOUT results.

Figure 7: Growth rate vs. q for geodesic acoustic mode model. The blue curve represents results from the 2DX code. The tan curve is an analytic solution for the GAM mode, whereas the red curve is an analytic solution for a zonal flow mode.

Figure 8: Growth rate vs. Z_{eff} for resistive ballooning mode in D-IIID edge geometry. These results are for mode number n=100. The black lines at the right hand side represent 2DX solutions for $\mu_{ii} = 0$ and $n = 10^2$,10^3, and 10^4. The dashed green line is an analytic solution assuming both Z_{eff} and mode

number are large.

Figure 9: Eigenmode structure from 2DX for resistive ballooning mode in D-IIID edge geometry. Dots indicate the positions of grid points in the 2DX mesh. Colors indicate absolute value of relative amplitude, whereas uncolored regions indicate values near zero.

Figure 10: Eigenmode structure from BOUT for resistive ballooning mode in D-IIID edge geometry. Dots indicate the positions of grid points in the BOUT mesh. Colors indicate absolute value of relative amplitude, whereas uncolored regions indicate values near zero.

Figure 11: Log-log plot of the relative error in the eigenvalues for a divertor-geometry solution of the resistive ballooning mode. Various resolutions for these cases with n x n grids are shown. The solid red line is a least-squares power law fit. The dashed black line is a 1/n power law fit.

Figure 12: Log-log plot of the (single processor) CPU time for various grid resolutions where $n = \sqrt{n_x n_y}$. The solid blue line is a least-squares power law fit to all the data. Red points indicate the square-grid runs. Green points indicate matrix set-up time which is more than an order of magnitude smaller than total computational time.

Figure 13: Growth rates as a function of mode number for LAPD eigenvalue scan. Red circles indicate 2DX results, blue circles indicate results from a 1D eigenmode solver.

Figure 14: Freqeuncies as a function of mode number for the LAPD eigenvalue scan. Red circles indicate 2DX results, blue circles indicate results from a 1D eigenmode solver.

Figure 15: Pressure profiles used in the 2DX/NSTX connection length study. The curves labeled a, b, and c represent different perturbed pressure profiles used to control the location of the dominant eigenmode.

Figure 16: Eigenmodes of the resistive ballooning model from the 2DX/NSTX connection length study. Red indicates high eigenmode amplitude, blue indicates low eigenmode amplitude. The eigenmodes a, b, and c are calculated using the corresponding pressure profiles from Fig. 15.

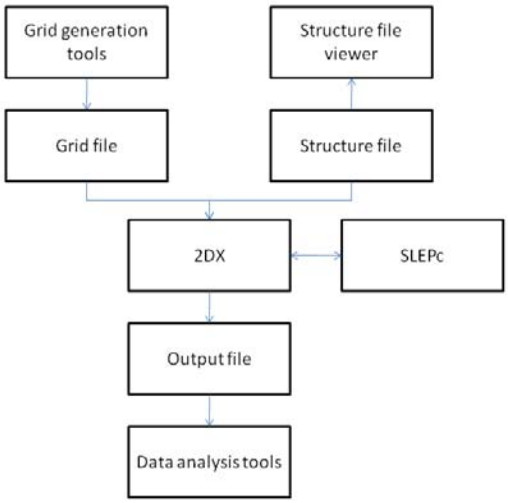
Grid generation tools
Structure file viewer
Grid file
Structure file
2DX
SLEPc
Output file
Data analysis tools

Figure 1:

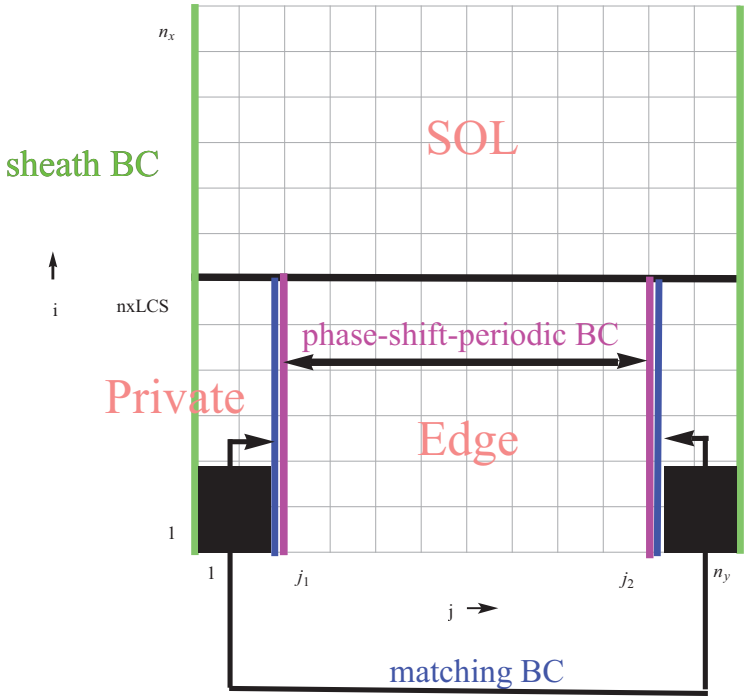

Figure 2:

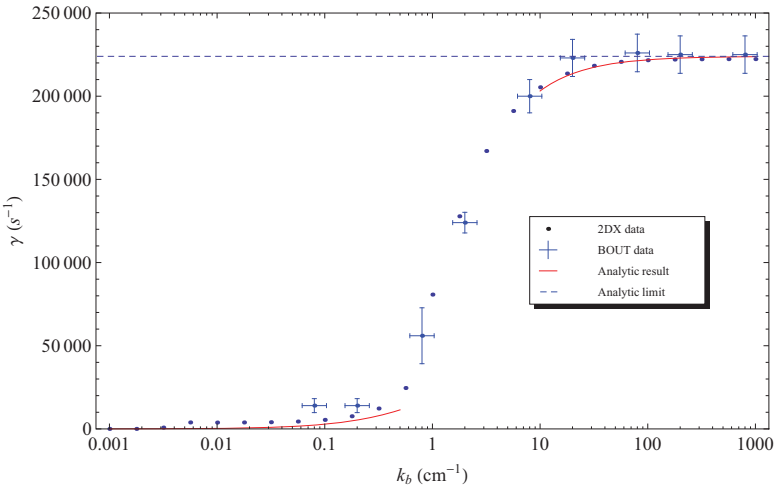

Figure 3:

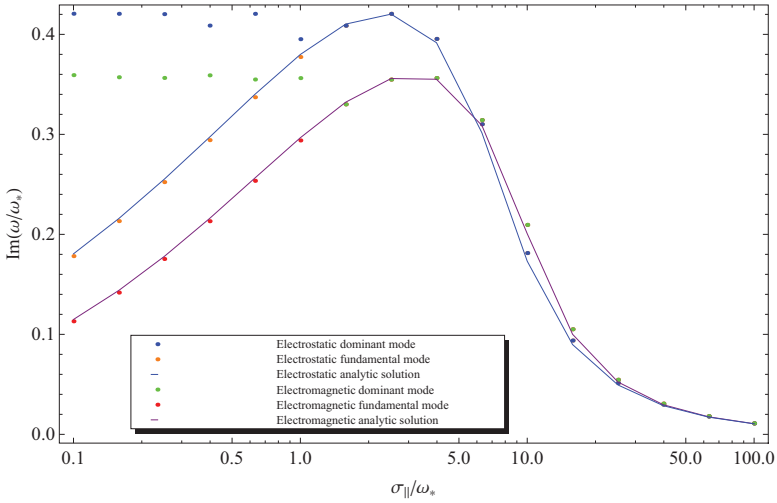

Figure 4:

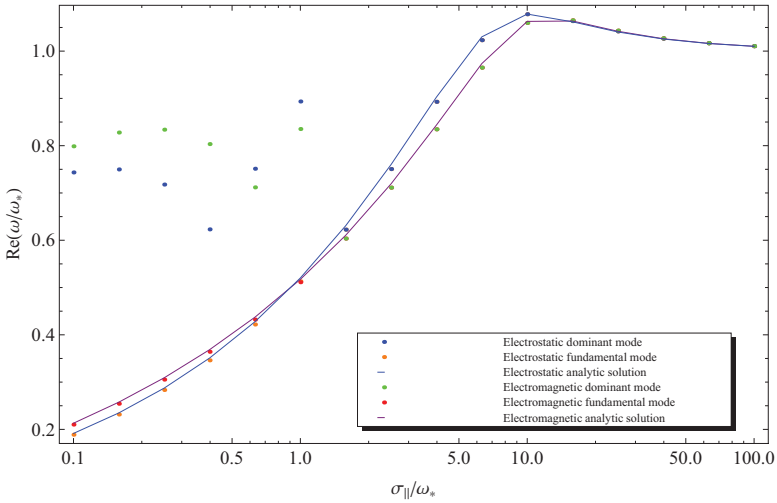

Figure 5:

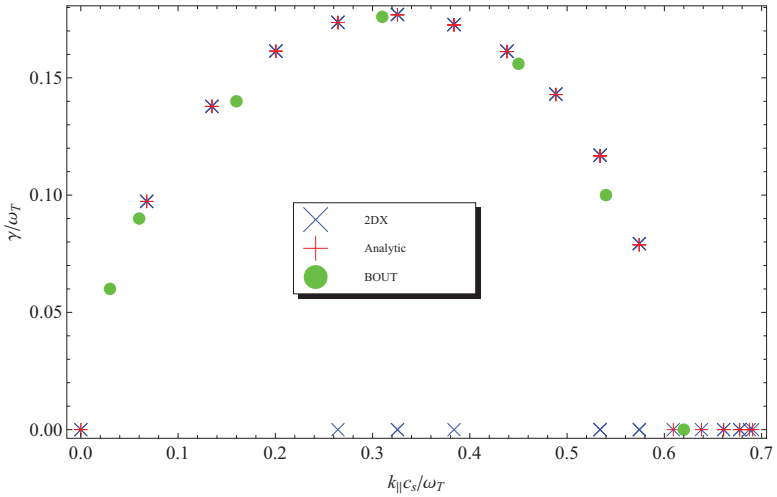

Figure 6:

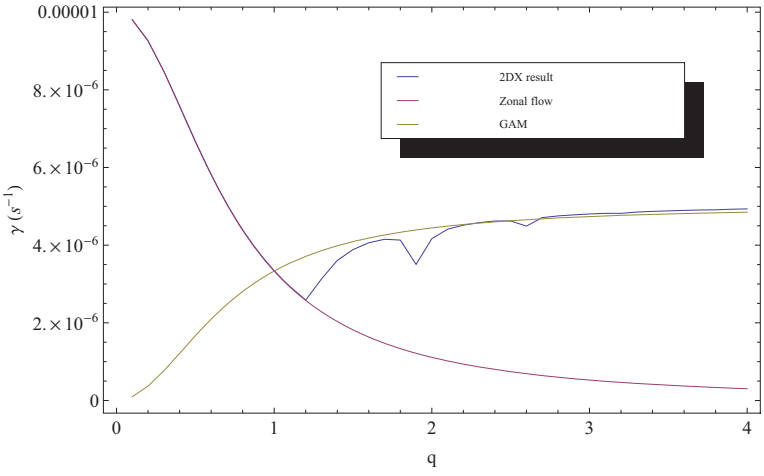

Figure 7:

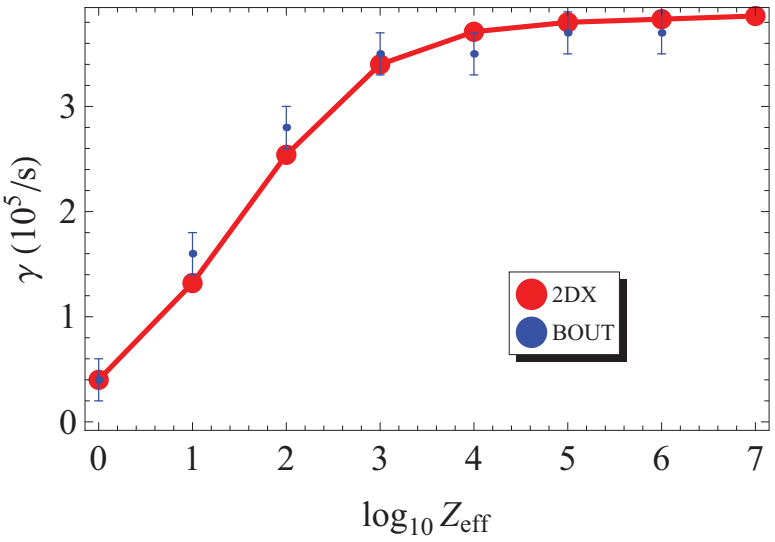

Figure 8:

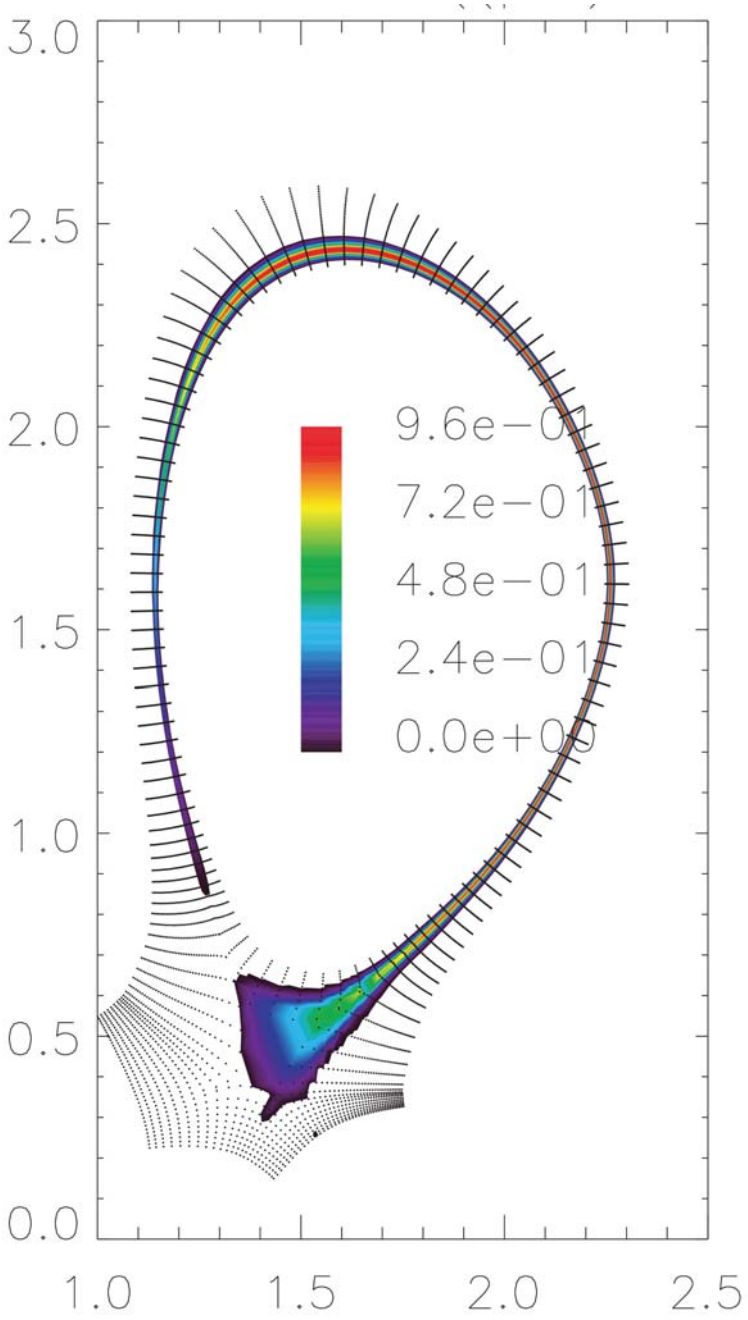

Figure 9:

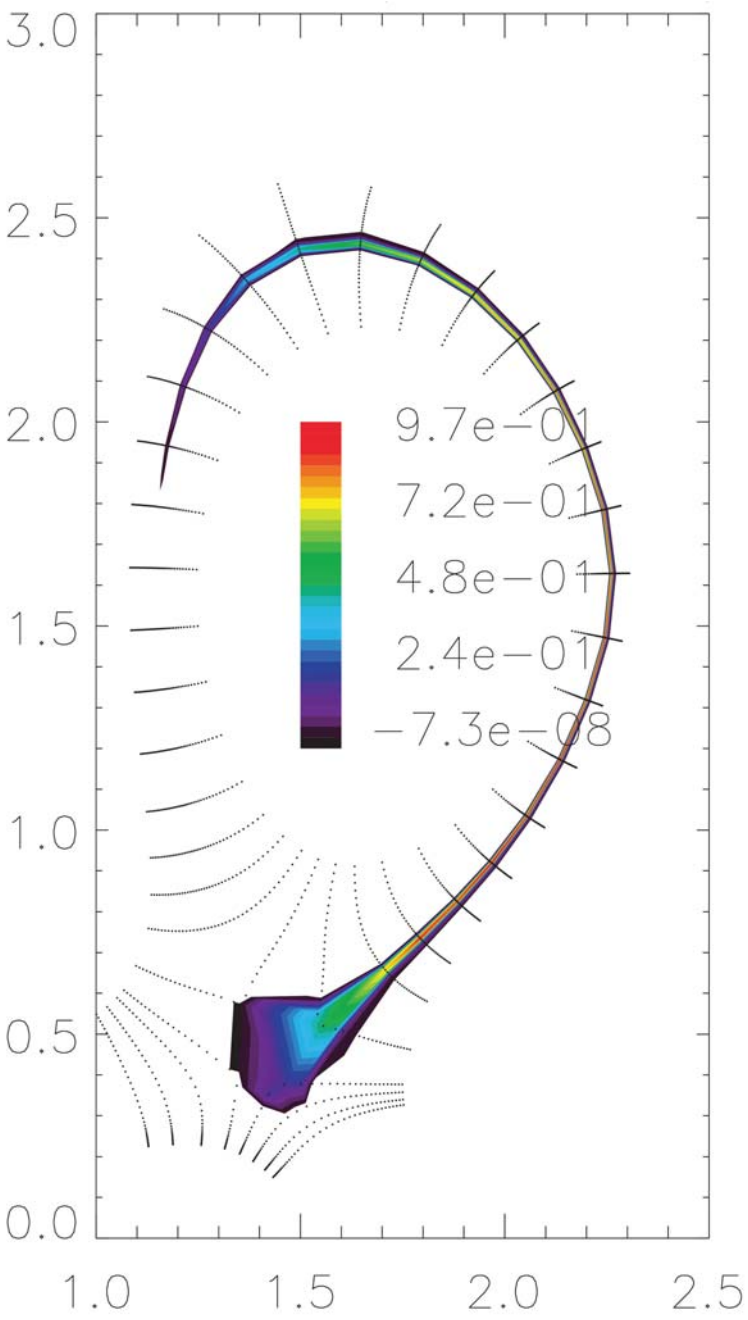

Figure 10:

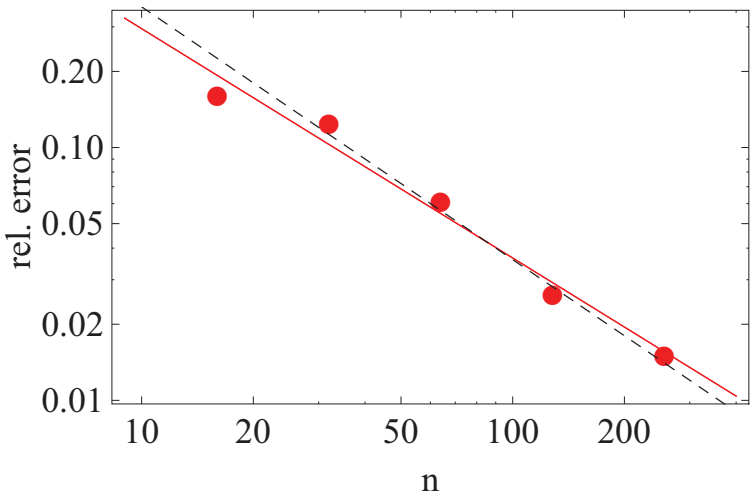

Figure 11:

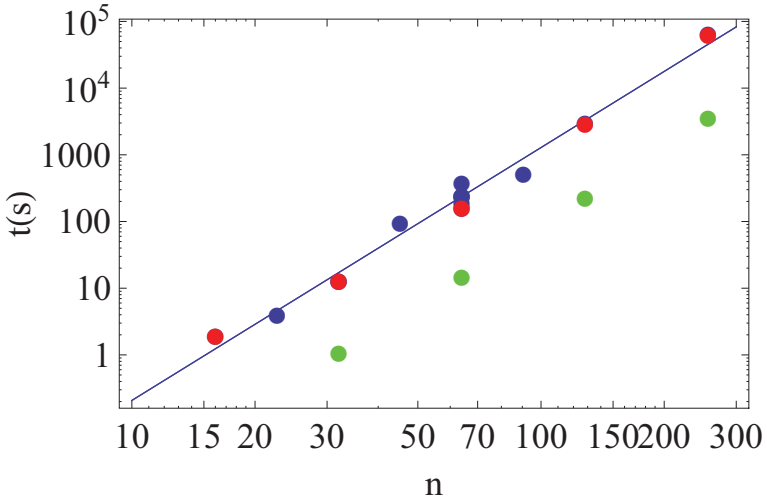

Figure 12:

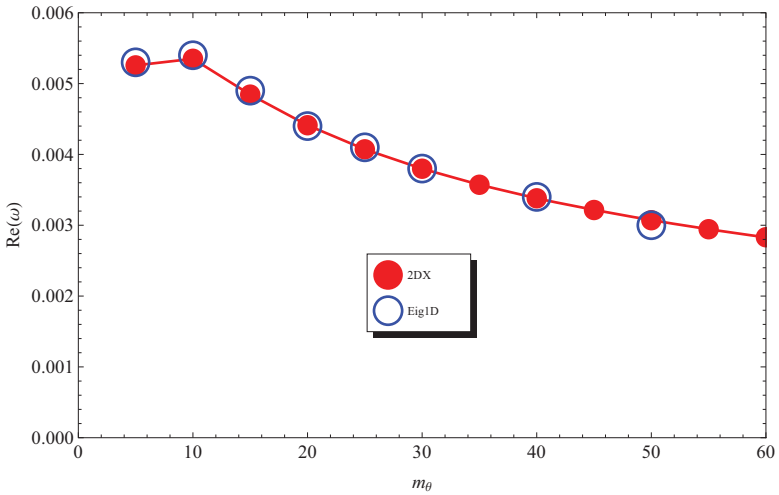

Figure 13:

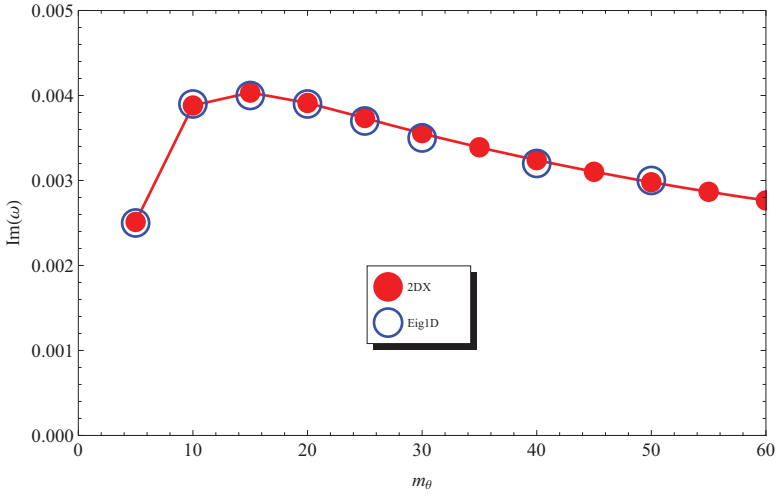

Figure 14:

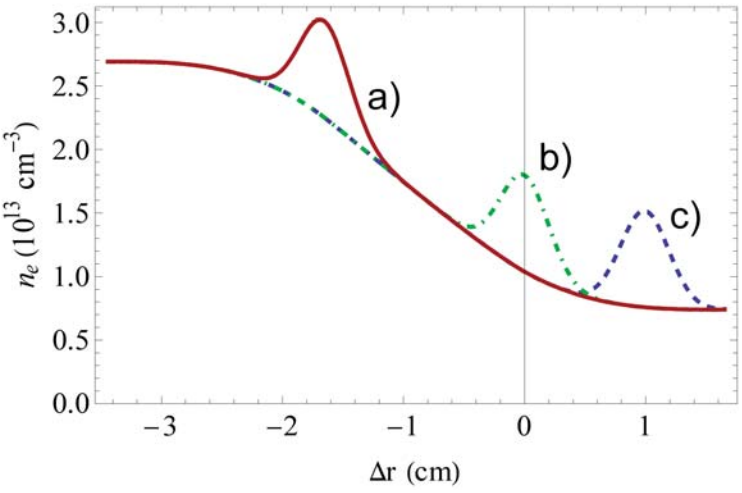

Figure 15:

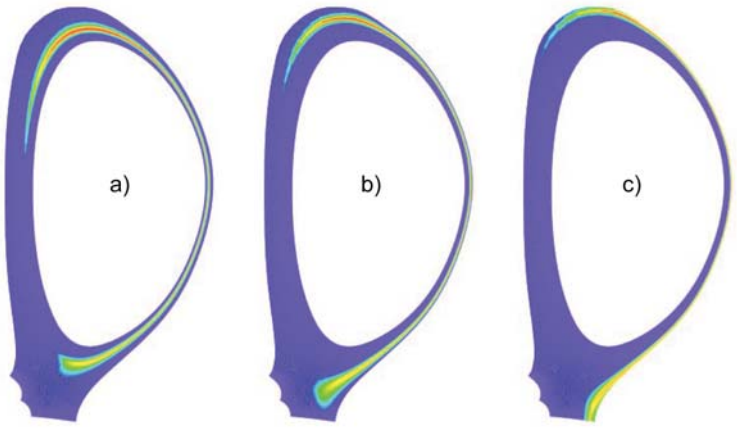

Figure 16: