testglf.tex
**The GLF23 Transport Model**
**Implemented by Jon Kinsey, General Atomics**

February 20, 2003

# 1 Introduction

This program is intended to serve as a stand-alone code for testing the GLF23 model for Ion Temperature Gradient (ITG) and Trapped Electron Mode (TEM) modes. The GLF23 model was formulated by approximating the linear growth rates of the 3D ballooning mode gyrokinetic (GKS) code whereby the transport coefficients were taken from simulations of a 3D nonlinear gyro-Landau-fluid (GLF) code [1]. The model contains magnetic shear and Shafranov shift ($\alpha$) stabilization in addition to $E \times B$ rotational shear stabilization. It is a comprehensive transport model that predicts particle, electron and ion thermal, toroidal momentum flows as well as turbulent electron–ion energy exchange.

It is a dispersion type transport model similar in construction to the fluid based Weiland ITG/TEM model where the diffusivities are found by solving the complex eigenvalue problem

$$Av = \lambda Bv$$

for a reduced set of perturbed equations of motion. Here, $\lambda = \hat{\omega} + i\hat{\gamma}$ is the eigenvalue and $v$ is the corresponding eigenvector. The current version of the model uses eight equations (**nroot = 8**) and is electrostatic. The user has the option to include impurity dynamics by setting **nroot = 12**, but this is usually a small effect at low to moderate values of $Z_{\rm eff}$. An internal eigenvalue solver (**cgg**) is incorporated inside the main subroutine **glf2d.f** utilizing a sequence of routines from the eispack package. The user has the option of using the **cgg** solver (default) by setting **leigen = 0** or using the more modern **tomsqz** solver (**leigen = 1**). The **tomsqz** routine has proven to be robust but solves the generalized eigenvalue problem which makes it more computationally intensive than the **cgg** eispack based solver. In our experience, the **cgg** solver has proven reliable on a number of platforms.

The eigenvalues yield the frequency and growth rates of the modes while the eigenvectors give the phase of the perturbed variables relative to one another. A nonlinear saturation rule is used to compute the transport for a spectrum of eigenmodes with 10 wavenumbers for the ion temperature gradient (ITG) and trapped electron modes (TEM) and 10 wavenumbers for the short wavelength electron temperature gradient (ETG) modes. A mixing length formula is used to give the heat diffusivity such that

$$\chi \sim 3/2(\gamma_{net}/k_x^2)\gamma_d\gamma/(\gamma^2 + \omega_o^2)$$

where $\omega_o$ is the mode frequency, $\gamma_d$ is radial mode damping rate, and $\gamma_{net} = \gamma - \gamma_E - \gamma_*$ with $\gamma$ denoting the mode growth rate in the absence of rotational shear and with $\gamma_E$ and $\gamma_*$ denoting the $E \times B$ and diamagnetic rotational shear rates, respectively.

# 2 Coordinate System for the Transport Model

It is assumed that profile information exists on a grid from 0 to **jmaxm** when the model is called. Gradients are computed with respect to the normalized square root of the toroidal flux $\hat{\rho} = \rho/\rho(a)$ with $\rho = [\Phi/(\pi B_T)]^{1/2}$ where $\Phi$ is the toroidal flux, $B_T$ is the toroidal field, and $a$ is the minor radius. For every unstable root in the $k_y$ spectrum, the quasi-linear diffusivities are derived summing over the individual contributions. The effective transport coefficients are in units of $m^2/s$ and are defined assuming that the the transport equations take the form prescribed by the ITER Expert Group,

$$\frac{3}{2}\frac{d(nT)}{dt} = \frac{1}{V'}\frac{d}{d\rho}\left[V'\left\langle|\nabla\rho|^2\right\rangle\chi\left(n\frac{dT}{d\rho}\right)\right] + \ldots$$

Here, $\chi$ is the effective diffusivity and no distinction is made between convection and conduction.
The toroidal momentum $\eta_\phi$ is defined with the momentum diffusion equation given as

$$\frac{d(Mn_iv_\phi)}{dt} = \frac{1}{V'}\frac{d}{d\rho}\left[V'\left\langle|\nabla\rho|^2\right\rangle\left(Mn_i\eta_\phi\frac{v_\phi}{d\rho} + Mv_\phi n_e\frac{dn_e}{d\rho}\right)\right]$$

## 3 Model Options for $E \times B$ shear stabilization

The original GLF23 model described in the 1996 Waltz paper [1] uses the flux–surface–averaged $E \times B$ shear rate $\gamma_E = (r/q)(\partial/\partial r)(qv_E/r)$ in circular geometry with a multiplier of $\alpha_e = 1.0$. Here, $v_E$ is the $E \times B$ shear velocity equal to $-E_r/B_\phi$. Since then it was discovered that the $E \times B$ rate can often exceed the maximum linear growth rate in the outer half of the plasma leading to the formation of an unphysical transport barrier. This was remedied by using a real geometry extension of $\gamma_E$ where the toroidal field in $v_E$ is replaced by $B_{\rm unit} = B_\phi \rho d\rho/rdr$ [2]. Here, the factor $\rho d\rho/rdr$ typically rises to 2–3 in the outer half of elongated plasmas, thus reducing the $E \times B$ rate in that region significantly. The corresponding multiplier for the original (retuned) models is $\alpha_e = 0.6(1.35)$ which has been found to be consistent with recent gyro–kinetic simulations by Waltz. To use the real geometry $E \times B$ shear, set **bt_flag=1** in the namelist input. The effective B–field is computed internally in subroutine callglf2d.f.

## 4 Retuning of GLF23

In the process of studying the effect of Shafranov shift stabilization in DIII–D and JET ITB discharges, an error was found in the $n = 0$ radial mode damping rate $\gamma_d$ which enters in the mixing length formula for the fluxes in the GLF23 model. An unphysical dependence on the MHD $\alpha$ in $\gamma_d$ through the diamagnetic drift frequency $\omega_D$ which resulted in the quasi-linear transport being periodic in $\alpha$. Once this was fixed it was discovered that the linear growth rates did not compare well with the linear gyrokinetic growth rates from GKS for weak or reversed magnetic shear parameters typically found within the deep core region of ITB discharges.

Recently, the fit formula for the extended ballooning mode angle has been retuned to give a better fit to the linear gyrokinetic growth rates for simulations including electrons. In particular, the magnetic shear and safety factor tuning parameters were changed and a new parameter for the MHD $\alpha$ was added. See the PDF file $gam_v s_s hat - std, NCS.pdf$ for a comparison of the growth rates versus magnet shear at safety factor values ranging from 1–5 for the standard test case at $\alpha = 0$ and for the NCS case at $\alpha = 1$ and $\alpha = 3$. The NCS parameters are the same as the standard parameters but have $a/L_{\rm Ti} = 10$, $a/L_{\rm Te} = 4$, and $\hat{s} = -0.5$. For the standard parameters, the retuned model gives roughly the same level of agreement as the original model. However, the retuned model yields much better agreement at weak and reversed magnetic shear for the NCS parameters. Near $\alpha = 4$ agreement with the GKS growth rates begins to be rapidly lost, so we have placed a limit on $\alpha$ within the code. If $\alpha$ exceeds 4.0, then $\alpha$ is set to 4.0.

Once the retuning of the growth rates was completed, the normalization of the diffusivities was performed using nonlinear GYRO ITG simulations assuming adiabatic electrons. See the accompanying PDF file $chii_v s_a/Lti - std.pdf$ which compares $\chi_i$ versus normalized ion temperature gradient using the original and retuned GLF23 models against the GYRO results. The renormalization results in a decrease in the coefficient for the ITG/TEM modes by a factor of 2 and in increase in the coefficient for the ETG modes of a factor of 3 with respect to the original GLF23 model. The norm factor for the ETG modes was determined using a small subset of DIII–D discharges. The radial mode damping exponent $adamp_g f$ was changed from 0.5 to 0.7 to give a better fit to the saturation at higher normalized temperature gradients. The $E \times B$ shear multiplier is $\alpha_e = 1.35$. To use the retuned version of the model, set **iglf=1**.

## 5 Initial Coding

The coding for the GLF23 driver begins as follows:

```
c@testglf.f
c 18-feb-03 version 1.6
c stand-alone driver for the GLF23 model
c written by Jon Kinsey, General Atomics
c---:----1----:----2----:----3----:----4----:----5----:----6----:----7-c
c
      implicit none
c
c...declare variables
c
      character cdate*24, line*132
```

```
c     character ctime*8  ! used if calling routine clock w/ +U77 option
      integer jpd

      double precision epsilon
      parameter ( jpd=50 )
c
      double precision te_m(0:jpd), ti_m(0:jpd)
     & , ne_m(0:jpd), ni_m(0:jpd), ns_m(0:jpd)
     & , zpte_m(0:jpd), zpti_m(0:jpd), zpne_m(0:jpd), zpni_m(0:jpd)
     & , angrotp_exp(0:jpd), egamma_exp(0:jpd), gamma_p_exp(0:jpd)
     & , vphi_m(0:jpd), vpar_m(0:jpd), vper_m(0:jpd)
     & , zeff_exp(0:jpd), bt_exp, bteff_exp(0:jpd), rho(0:jpd), arho_exp
     & , gradrho_exp(0:jpd), gradrhosq_exp(0:jpd)
     & , rmin_exp(0:jpd), rmaj_exp(0:jpd), rmajor_exp
     & , q_exp(0:jpd), shat_exp(0:jpd), alpha_exp(0:jpd)
     & , elong_exp(0:jpd), zimp_exp, amassimp_exp, amassgas_exp
     & , alpha_e, x_alpha
c
      double precision zpte_in, zpti_in, zpne_in, zpni_in, drho
c
      double precision diffnem, chietem, chiitim
     & , etaphim, etaparm, etaperm, exchm
     & , diff_m(0:jpd), chie_m(0:jpd), chii_m(0:jpd), etaphi_m(0:jpd)
     & , etapar_m(0:jpd), etaper_m(0:jpd), exch_m(0:jpd)
     & , egamma_m(0:jpd), egamma_d(0:jpd,10), gamma_p_m(0:jpd)
     & , anrate_m(0:jpd), anrate2_m(0:jpd)
     & , anfreq_m(0:jpd), anfreq2_m(0:jpd)
c
      integer lprint, nroot, jshoot, jmm, jmaxm, itport_pt(1:5)
     & , igrad, idengrad, i_delay, j, k, leigen, irotstab, bt_flag, iglf
```

The variables in the input file **in** are read through the namelist **nlglf**:

```
c
      namelist /nlglf/ leigen, lprint, nroot, iglf, jshoot, jmm, jmaxm
     & , itport_pt, irotstab, te_m, ti_m, ne_m, ni_m, ns_m
     & , igrad, idengrad, zpte_in, zpti_in, zpne_in, zpni_in
     & , angrotp_exp, egamma_exp, gamma_p_exp, vphi_m, vpar_m, vper_m
     & , zeff_exp, bt_exp, bt_flag, rho, arho_exp
     & , gradrho_exp, gradrhosq_exp
     & , rmin_exp, rmaj_exp, rmajor_exp, q_exp, shat_exp
     & , alpha_exp, elong_exp, zimp_exp, amassimp_exp, amassgas_exp
     & , alpha_e, x_alpha
c
```

# 6 Input and Output Variables

The following two tables list the input and output variables for the GLF23 model. Internal documentation can also be found inside the callglf2d and glf2d routines. The variable **jshoot** determines how the gradients and diffusivities are computed on the grid where jm is the local grid point and is between 0 and the maximum grid point **jmaxm**. The choice of **jshoot** depends upon the structure of the transport code and/or user preference. Typically, for a time-dependent transport code, **jshoot=0** so the gradients and diffusion are computed on the backward gradient between the jm+1 and jm grid points. The model can also be run inside a shooting code (**jshoot=1**) whereby the diffusivities at grid point jm are computed on the forward (implicit) gradient between jm to jm-1. To turn on the impurity dynamics, set **idengrad=3** and prescribe the fast ion density **ns_m** along with the average mass **amass_imp** and charge **zimp_exp** of the impurity. The default is for simple dilution (**idengrad=2**). The gradients can be prescribed externally using the variables **zpmte**, etc. and by setting **igrad=1**. If the user chooses to prescribe the gradients, which are scalar quantities, then the user must also supply the corresponding grid number **jmm**. In this case, the output scalar variables are used **diffnem**, etc.

**Inputs:**

| variable | default | symbol | meaning |
|---|---|---|---|
| `leigen` | 0 | | eigenvalue solver, default 1 for tomsqz, =0 for cgg solver |
| `jshoot` | 0 | | = 0 for time-dep code, = 1 for shooting code |
| `iglf` | 0 | | = 0 for original normalization, = 1 retuned model |
| `jmm` | 0 | | grid number (jmm=0 does full grid from jm=1 to jmaxm-1) |
| `jmaxm` | 0 | | maximum no. of grid pts. All input profiles are dimensioned (0:jmaxm) |
| `itport_pt(1:5)` | 0 | | controls transport, = 1 for on. |
| | 0 | itport_pt(1) | density transport |
| | 0 | itport_pt(2) | electron transport |
| | 0 | itport_pt(3) | ion transport |
| | 0 | itport_pt(4) | vphi transport (-1 use egamma_exp) |
| | 0 | itport_pt(5) | vtheta transport (-1 use gamma_p_exp) |
| `te_m(jm)` | 1.0 | $T_e$ | electron temperature (keV) |
| `ti_m(jm)` | 1.0 | $T_i$ | Ion temperature (keV) |
| `ne_m(jm)` | 1.0 | $n_e$ | electron density $(10^{19}m^{-3})$ |
| `ni_m(jm)` | 1.0 | $n_i$ | Ion density $(10^{19}m^{-3})$ |
| `ns_m(jm)` | 1.0 | $n_s$ | Fast ion density $(10^{19}m^{-3})$ |
| `igrad` | 0 | | compute gradients (1=input gradients) |
| `idengrad` | 2 | | compute simple dilution $dil_gf = 1 - n_i/n_e$ (3=actual dilution) |

**Inputs:**

| variable | default | symbol | meaning |
|---|---|---|---|
| `zpte_in` | 0.0 | | log electron temperature gradient with respect to $\rho$ |
| `zpti_in` | 0.0 | | log ion temperature gradient |
| `zpne_in` | 0.0 | | log electron density gradient |
| `zpni_in` | 0.0 | | log ion density gradient |
| `angrotp_exp(jm)` | 0.0 | | experimental toroidal angular velocity (1/s) |
| `egamma_exp(jm)` | 0.0 | $\gamma_e^{\rm exp}$ | experimental $E \times B$ shearing rate |
| `gamma_p_exp(jm)` | 0.0 | $\gamma_p^{\rm exp}$ | experimental parallel velocity shearing rate |
| `vphi_m(jm)` | 0.0 | $v_\phi$ | toroidal velocity (m/s) |
| `vpar_m(jm)` | 0.0 | $v_\parallel$ | parallel velocity (m/s) |
| `vper_m(jm)` | 0.0 | $v_\perp$ | perpendicular velocity (m/s) |
| `zeff_exp(jm)` | 1.0 | $Z_{\rm eff}$ | effective charge |
| `bt_exp` | 1.0 | $B_{\rm T}$ | toroidal field (T) such that $B_T \pi \rho(a)^2$=enclosed flux |
| `bt_flag` | 0.0 | | switch for effective B–field ($> 0$ for $B_{\rm eff}$, $B_{\rm T}$ otherwise) |
| `rho(jm)` | 0.0 | $\hat{\rho}$ | square root of normalized toroidal flux surface label |
| `arho_exp` | 1.0 | $\rho_a$ | square root of toroidal flux at last closed flux surface LCFS (m) |
| `gradrho_exp(jm)` | 1.0 | $< \|\nabla\rho\| >$ | dimensionless |
| `gradrhosq_exp(jm)` | 1.0 | $< \|\nabla\rho\|^2 >$ | dimensionless |
| `rmin_exp(jm)` | 0.0 | $r_{\rm min}$ | local minor radius (m) |
| `rmaj_exp(jm)` | 0.0 | $r_{\rm maj}$ | local major radius (m) |
| `rmajor_exp` | 1.0 | $R_{\rm o}$ | geometrical major radius of magnetic axis (m) |
| `q_exp(jm)` | 1.0 | $q$ | safety factor |
| `shat_exp(jm)` | 0.0 | $\hat{s}$ | magnetic shear $d \ln q / d \ln \rho$ |
| `alpha_exp(jm)` | 0.0 | $\alpha$ | MHD alpha $-q^2 R (d\beta/dr)$ |
| `elong_exp(jm)` | 1.0 | $\kappa$ | local elongation |
| `amassgas_exp` | 1.0 | $M_H$ | atomic number of working gas |
| `amassimp_exp` | 12.0 | $M_Z$ | average mass of impurity |
| `zimp_exp` | 6.0 | $Z_{\rm imp}$ | average charge of impurity |
| `alpha_e` | 0.0 | $\alpha_E$ | real variable switch for $E \times B$ shear stabilization (0 = off and ¿ 0 = on) |
| `x_alpha` | 0.0 | $x_\alpha$ | real variable switch for alpha stabilization (0 = off, 1. = on and -1. for self-consistent alpha stabilization) |
| `i_delay` | 0 | | time steps by which $E \times B$ shear *egamma_d* is delayed (for numerical stability and normally should be defaulted) |

Note: the input file **in** is copied to a file called **temp** whereby any comments are stripped out using subroutine **stripx**.

**Output:**

| variable | default | symbol | meaning |
|---|---|---|---|
| **diffnem** | 0.0 | $D$ | ion plasma diffusivity ($m^2/s$) |
| **chienem** | 0.0 | $\chi_e$ | electron thermal diffusivity ($m^2/s$) |
| **chiinem** | 0.0 | $\chi_i$ | ion thermal diffusivity ($m^2/s$) |
| **etaphim** | 0.0 | $\eta_\phi$ | toroidal velocity diffusivity ($m^2/s$) |
| **etaparm** | 0.0 | $\eta_\parallel$ | parallel velocity diffusivity ($m^2/s$) |
| **etaperm** | 0.0 | $\eta_\perp$ | perpendicular velocity diffusivity ($m^2/s$) |
| **exchm** | 0.0 | | turbulent electron-ion energy exchange ($MW/m^3$) |
| **diff_m(jm)** | 0.0 | $D$ | ion plasma diffusivity ($m^2/s$) |
| **chie_m(jm)** | 0.0 | $\chi_e$ | electron thermal diffusivity ($m^2/s$) |
| **chii_m(jm)** | 0.0 | $\chi_i$ | ion thermal diffusivity ($m^2/s$) |
| **etaphi_m(jm)** | 0.0 | $\eta_\phi$ | toroidal velocity diffusivity ($m^2/s$) |
| **etapar_m(jm)** | 0.0 | $\eta_\parallel$ | parallel velocity diffusivity ($m^2/s$) |
| **etaper_m(jm)** | 0.0 | $\eta_\perp$ | perpendicular velocity diffusivity ($m^2/s$) |
| **exch_m(jm)** | 0.0 | | turbulent electron-ion energy exchange ($MW/m^3$) |
| **egamma_m(jm)** | 0.0 | $\gamma_e$ | $E \times B$ shear rate in units of csda_m (1/s) |
| **egamma_d(jm)** | 0.0 | $\gamma_e^d$ | $E \times B$ shear rate delayed by i_delay steps |
| **gamma_p_m(jm)** | 0.0 | $\gamma_p$ | parallel velocity shear rate in units of local csda_m (1/s) |
| **anrate_m(jm)** | 0.0 | $\gamma_1$ | growth rate of leading mode in units of local csda_m (1/s) |
| **anrate2_m(jm)** | 0.0 | $\gamma_2$ | 2nd leading growth rate in units of local csda_m (1/s) |
| **anfreq_m(jm)** | 0.0 | $\omega_1$ | leading mode frequency in units of local csda_m (1/s) |
| **anfreq2_m(jm)** | 0.0 | $\omega_2$ | 2nd leading mode frequency in units of local csda_m (1/s) |

Note: if input variable **jmm** is set to a non-zero value, the scalar output quantities (**diffnem** et al.) are used. If **jmm** is zero, the profile output arrays (**diff_m** et al.) are used. If the user supplies scalar input gradients, then a non-zero **jmm** value must be used.

Initialization of input and output variables:

```
c---:----1----:----2----:----3----:----4----:----5----:----6----:----7-c
      open (4,file='temp')
      open (5,file='in')
      open (6,file='out')
c
      call stripx (5,4,6)
c
      cdate = ' '
c     ctime = ' '
c
      call c9date (cdate)
cray      call clock (ctime)
cibm      call clock_ (ctime)
c
      write (6,*)
      write (6,*) ' GLF23 stand-alone code by Kinsey, GA  ',cdate
c
c..default inputs
c
      epsilon  = 1.e-10
      leigen   = 0   ! for cgg eigenvalue solver
      nroot    = 8   ! number of roots in eigenvalue solver
      iglf     = 0   ! original GLF23 normalization
      jshoot   = 0   ! for time-dependent code
      jmaxm    = 2
      igrad    = 0   ! compute gradients
      idengrad = 2   ! simple dilution
```

```
      i_delay  = 0
      itport_pt(1) = 0
      itport_pt(2) = 0
      itport_pt(3) = 0
      itport_pt(4) = 0
      itport_pt(5) = 0
      irotstab     = 1    ! use internally computed ExB shear, 0 for prescribed
      bt_exp       = 1.0
      bt_flag      = 0    ! do not use effective B-field
      bteff_exp    = 1.0  ! effective B-field (used when bt_flag > 0)
      rmajor_exp   = 1.0
      amassgas_exp = 1.0
      zimp_exp     = 6.0
      amassimp_exp = 12.0
      arho_exp     = 1.0
      alpha_e      = 0.   ! ExB shear stabilization
      x_alpha      = 0.   ! alpha stabilization
      zpte_in      = 0.
      zpti_in      = 0.
      zpne_in      = 0.
      zpni_in      = 0.
c
      do j=0,jpd
        te_m(j)   = 0.0
        ti_m(j)   = 0.0
        ne_m(j)   = 0.0
        ni_m(j)   = 0.0
        ns_m(j)   = 0.0
c
        zpte_m(j) = 0.0
        zpti_m(j) = 0.0
        zpne_m(j) = 0.0
        zpni_m(j) = 0.0
c
        angrotp_exp(j)   = 0.0
        egamma_exp(j)    = 0.0
        gamma_p_exp(j)   = 0.0
        vphi_m(j)        = 0.0
        vpar_m(j)        = 0.0
        vper_m(j)        = 0.0
c
        zeff_exp(j)   = 1.0
        rho(j)        = 0.0
        gradrho_exp(j)   = 1.0
        gradrhosq_exp(j) = 1.0
        rmin_exp(j)   = 0.0
        rmaj_exp(j)   = 0.0
        q_exp(j)      = 1.0
        shat_exp(j)   = 0.0
        alpha_exp(j)  = 0.0
        elong_exp(j)  = 1.0
      enddo
c
c..default outputs
c
      diffnem      = 0
      chietem      = 0
      chiitim      = 0
      etaphim      = 0
      etaparm      = 0
      etaperm      = 0
```

```
      exchm        = 0
c
      do j=0,jpd
        diff_m(j)    = 0.0
        chie_m(j)    = 0.0
        chii_m(j)    = 0.0
        etaphi_m(j) = 0.0
        etapar_m(j) = 0.0
        etaper_m(j) = 0.0
        exch_m(j)    = 0.0
        egamma_m(j) = 0.0
        gamma_p_m(j)= 0.0
        anrate_m(j) = 0.0
        anrate2_m(j) = 0.0
        anfreq_m(j) = 0.0
        anfreq2_m(j) = 0.0
        do k=1,10
          egamma_d(j,k) = 0.0
        enddo
      enddo
```

Read input file and copy to output file:

```
c
c---:----1----:----2----:----3----:----4----:----5----:----6----:----7-c
c
c..read input file
c
  10  continue
c
      read  (4,20,end=900,err=900) line
  20  format (a)
c
      if ( index ( line, '$nlglf' ) .gt. 0
     &    .or. index ( line, '&nlglf' ) .gt. 0 ) then
c
c  read namelist input
c
        backspace 4
        read  (4,nlglf)
      else
        go to 10
      endif
c
      if ( leigen .gt. 0 ) then
        write (6,*) ' tomsqz eigenvalue solver '
      else
        write (6,*) ' cgg eigenvalue solver '
      endif
c
      if ( lprint .gt. 100 ) write (6,nlglf)
c
      write (6,*)
```

The pre-call subroutine **callglf2d** prescribes the GLF23 model parameters, sets up gradients, $E \times B$ shearing rate, etc. and then calls the main routine **glf2d** containing the GLF23 model. Unlike **callglf2d** which is accessed through an argument list, information is passed to and from **glf2d** through a local common block **glf.m**. Upon returning from the **glf2d** routine, the diffusivities, exchange terms, growth rates and frequencies are determined.

```
c
        call callglf2d( leigen, nroot, iglf
     & , jshoot, jmm, jmaxm, itport_pt
     & , irotstab, te_m, ti_m, ne_m, ni_m, ns_m
```

```
     & , igrad, idengrad, zpte_in, zpti_in, zpne_in, zpni_in
     & , angrotp_exp, egamma_exp, gamma_p_exp, vphi_m, vpar_m, vper_m
     & , zeff_exp, bt_exp, bt_flag, rho
     & , arho_exp, gradrho_exp, gradrhosq_exp
     & , rmin_exp, rmaj_exp, rmajor_exp, zimp_exp, amassimp_exp
     & , q_exp, shat_exp, alpha_exp, elong_exp, amassgas_exp
     & , alpha_e, x_alpha, i_delay
     & , diffnem, chietem, chiitim, etaphim, etaparm, etaperm
     & , exchm, diff_m, chie_m, chii_m, etaphi_m, etapar_m, etaper_m
     & , exch_m, egamma_m, egamma_d, gamma_p_m
     & , anrate_m, anrate2_m, anfreq_m, anfreq2_m )
c
      do j=1,jmaxm
        drho=rho(j-1)-rho(j)+epsilon
        zpte_m(j)=-(log(te_m(j-1))-log(te_m(j)))/drho
        zpti_m(j)=-(log(ti_m(j-1))-log(ti_m(j)))/drho
        zpne_m(j)=-(log(ne_m(j-1))-log(ne_m(j)))/drho
        zpni_m(j)=-(log(ni_m(j-1))-log(ni_m(j)))/drho
      enddo
```

# 7 Printout

Relevant printout of the input and output variables including the diffusivites and the growth rates and frequencies of the first and second roots from the dispersion relation are given below.
Inside the **glf2d** routine, diagnostic printout is provided if the variable **lprint** is greater than zero. Setting **lprint** to 6 prints the matrices and solution to the eigenvalue equation and setting it higher to 99 gives detailed output from the model. No diagnostic printout is given inside routine **callglf2d**.

```
c
c..printout
c

      if ( lprint .gt. 0 ) then
        write(6,100)
        do j=0,jmaxm
          write (6,110) rho(j), te_m(j), ti_m(j), ne_m(j), ni_m(j)
     & ,        zeff_exp(j), q_exp(j), shat_exp(j)
        enddo
c
        write(6,120)
        do j=0,jmaxm
          write(6,110) rho(j), zpte_m(j), zpti_m(j)
     & ,       zpne_m(j), zpni_m(j)
        enddo
c
        write(6,130)
        do j=0,jmaxm
          write (6,110) rho(j), diff_m(j), chie_m(j), chii_m(j)
     & ,        etaphi_m(j), etapar_m(j), etaper_m(j), exch_m(j)
        enddo
c
        write(6,140)
        do j=0,jmaxm
          write(6,110) rho(j), egamma_m(j), gamma_p_m(j)
     & ,       anrate_m(j),anrate2_m(j), anfreq_m(j), anfreq2_m(j)
        enddo
      endif
c
 100    format(t5,'rho',t13,'Te',t21,'Ti',t29,'ne',t37,'ni'
     & ,      t44,'Zeff',t53,'q',t60,'shear',t68,'#prof')
 110    format (11(0pf8.4))
```

```
 120    format(/,t5,'rho',t12,'zpte',t20,'zpti',t28,'zpne'
     & ,      t36,'zpni',t68,'#log-grad')
 130    format(/,t5,'rho',t12,'diff',t20,'chie',t28,'chii'
     & ,      t35,'etaphi',t43,'etapar',t51,'etaper',t60
     & ,      'exch',t68,'#chi')
 140    format(/,t5,'rho',t10,'egamma_m',t19, 'gamma_p',t27
     & ,      'anrate', t35,'anrate2',t43,'anfreq',t51,'anfreq2'
     & ,      t68,'#gamma')
c
 900  continue
c
      stop
      end
```

# 8 Sample test cases

Four test cases were prepared for testing of the GLF23 model. These cases were constructed using data for the low $\rho_*$ DIII-D discharge #82205 taken from the ITER Profile Database. The input file **in-1** calls the original model and gives the profile information at 4 radial points around $\rho = 0.5$ with $E \times B$ shear turned off and Shafranov–shift ($\alpha$) stabilization turned on. Given below is the output from the first test case. The second input file **in-2** is the same except that $E \times B$ shear stabilization is turned on in the model using the circular formula given in original Phys. Plasmas paper by Waltz, et al. [1]. The third input file **in-3** is for the original GLF23 model with real geometry $E \times B$ shear and a multiplier of 0.6. The fourth input file **in-4** is the same as in the third test case only using the retuned GLF23 model (iglf=1) with $\alpha_e = 1.35$.

```
 &nlglf
 ! GLF23 test case 10/12/98
 ! DIII-D #82205 at rho=0.5 via ITER PDB
 ! original GLF23 model
 ! ExB shear off, alpha-stabilization on
 ! only electron and ion thermal transport
 leigen = 0,
 lprint = 1,
 jmm = 0,
 jmaxm = 3,
 jshoot = 1,
 igrad = 0,
 itport_pt(1) = 0,
 itport_pt(2) = 1,
 itport_pt(3) = 1,
 itport_pt(4) = 0,
 itport_pt(5) = 0,
 rho  = 4.80000E-01,5.00000E-01,5.20000E-01,5.40000E-01,
 ne_m = 5.19480E+00,5.06580E+00,4.94040E+00,4.81920E+00,
 ni_m = 4.59805E+00,4.48724E+00,4.37882E+00,4.27359E+00,
 te_m = 2.19700E+00,2.12590E+00,2.05940E+00,1.99690E+00,
 ti_m = 2.93110E+00,2.83010E+00,2.73050E+00,2.63240E+00,
 zeff_exp    = 1.37000E+00,1.37000E+00,1.37000E+00,1.37000E+00,
 angrotp_exp = 6.50101E+04,6.54766E+04,6.34395E+04,6.19369E+04,
 egamma_exp  = 2.44646E-02,3.71168E-02,6.94274E-02,9.87615E-02,
 gamma_p_exp = 0.00,0.00,0.00,0.00,
 vphi_m = 0.00,0.00,0.00,0.00,
 vpar_m = 0.00,0.00,0.00,0.00,
 vper_m = 0.00,0.00,0.00,0.00,
 bt_exp   = 1.79825E+00,
 arho_exp = 8.16454E-01,
 gradrho_exp   = 1.07161E+00,1.07687E+00,1.08273E+00,1.08909E+00,
 gradrhosq_exp = 1.18644E+00,1.20134E+00,1.21818E+00,1.23717E+00,
 rmin_exp  = 3.41280E-01,3.54740E-01,3.68220E-01,3.81520E-01,
 rmaj_exp  = 1.72940E+00,1.72750E+00,1.72560E+00,1.72380E+00,
 rmajor_exp= 1.68367E+00,
```

```
q_exp     = 1.33110E+00,1.37380E+00,1.42250E+00,1.47770E+00,
shat_exp  = 6.69689E-01,7.73522E-01,8.88209E-01,1.00876E+00,
alpha_exp = 4.52236E-01,4.62719E-01,4.65413E-01,4.77653E-01,
elong_exp = 1.30650E+00,1.31050E+00,1.31460E+00,1.31940E+00,
amassgas_exp = 2.00000E+00,
alpha_e = 0.0,
bt_flag = 0,
x_alpha = 1.0,
/

 GLF23 stand-alone code by Kinsey, GA  19feb2003
 cgg eigenvalue solver

   rho     Te      Ti      ne      ni     Zeff     q      shear   #prof
 0.4800  2.1970  2.9311  5.1948  4.5980  1.3700  1.3311  0.6697
 0.5000  2.1259  2.8301  5.0658  4.4872  1.3700  1.3738  0.7735
 0.5200  2.0594  2.7305  4.9404  4.3788  1.3700  1.4225  0.8882
 0.5400  1.9969  2.6324  4.8192  4.2736  1.3700  1.4777  1.0088

   rho    zpte    zpti    zpne    zpni                            #log-grad
 0.4800  0.0000  0.0000  0.0000  0.0000
 0.5000  1.6449  1.7533  1.2573  1.2197
 0.5200  1.5890  1.7914  1.2533  1.2229
 0.5400  1.5409  1.8294  1.2419  1.2163

   rho    diff    chie    chii   etaphi  etapar  etaper   exch    #chi
 0.4800  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
 0.5000  0.0000 79.9161 66.0614  2.4782  0.0001 16.5789 -0.2853
 0.5200  0.0000 56.8475 47.9021  2.1036  0.0000 14.0228 -0.2490
 0.5400  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000

   rho  egamma_m gamma_p anrate  anrate2 anfreq  anfreq2          #gamma
 0.4800  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
 0.5000 -0.0148 -0.0025  0.1216  0.1081 -0.2785 -0.1472
 0.5200 -0.0039 -0.0018  0.0908  0.0851 -0.1892 -0.3402
 0.5400  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
```

# References


[1] R. E. Waltz, G. M. Staebler, W. Dorland, G. W. Hammett, M. Kotschenreuther, and J. A. Konings, Phys. Plasmas **4**, 2482 (1997).

[2] R. E. Waltz and R. L. Miller, Phys. Plasmas **6**, 4265 (1999).