How to run QuaLiKiz

January 30, 2017

Abstract

This manual explains how to build QuaLiKiz and set up standalone scans. The inputs and outputs are also detailed. The standalone version is run from within a MATLAB environment. There is also the possibility of launching a run directly from CRONOS data. This feature is explained. Scripts for building QuaLiKiz on either the CEA Zone Partenaires Andromede cluster, the Gateway and JET JAC cluster are all available. Any questions can be sent to Jonathan Citrin at J.Citrin@differ.nl

1 Code capabilities

QuaLiKiz first calculates the dispersion relation to solve the linear gyrokinetic equation for instabilities in the ITG/TEM and ETG scales. Then, based on the dispersion relation solution, the quasilinear fluxes are calculated. Single-scale saturation rules are employed separately for the ion and electron scales. It is also possible to run only-ion-scale or only-electron scale calculations.

QuaLiKiz can calculate particle, heat, and momentum transport for electrons and an arbitrary number of ion species. Electrons can be kinetic or adiabatic. Ions can be kinetic or 'trace'. Trace ions are not included in the dispersion relation calculation but their fluxes are calculated. Ions are always considered adiabatic for electron scale calculations ($k_\theta \rho_i > 2$).

The impact of rotation and rotation shear is included. This includes rotodiffusion (important for heavy impurities), $E \times B$ stabilization, and Coriolis pinch. In addition, the impact of poloidal asymmetries due to centrifugal forces and/or temperature anisotropies are included. This is important for heavy impurity transport.

Approximate calculation times to be expected are 1 CPU s for a single wavenumber. This increases to ≈ 5 CPU s when rotation is included in the dispersion relation (due to losing symmetry in the integrals). The code is parallelized with a hybrid MPI and OpenMP scheme.

2 Compiling the code

From the src directory, run the relevant script for your environemnt, i.e. either type './makeandromede.sh', './makegateway.sh' or './makejac.sh'. The Qua-

LiKiz.exe executable will be created in the src directory. A NAG installation is necessary. Apart from NAG, no other libraries need to be loaded and the code is self-contained. This facilitates porting to other systems.

3 Setting up a new run

Enter a MATLAB environment. From the main QuaLiKiz directory first run the script newrun.m. When prompted, type in the name of your new run, e.g. 'run10_ILOVEFUSION'. This directory will be created in the 'runs' directory. If this directory does not already exist it will be created.

You will now find yourself in this directory, together with a bunch of m-files copied from the 'tools' directory. These m-files will be needed to create a new QLK run as described below and will be introduced where relevant in the following sections.

3.1 Setting up a new scan

A template input file - parameters.m - has been copied to the working directory. All the external parameters needed for a QLK scan can be found and modified here. In general, 2D scans are carried out. One dimension is k_θ, while all other variables (e.g. magnetic shear, R/L_{Ti}) are of size scann (size of the second dimension) and can be either constant in that dimension, or varying if that parameter is the subject of the scan. ktheta_min, ktheta_max, size of the ktheta grid, and the value of scann are all variables in the parameter file. In the default parameters.m example, the magnetic shear is the subject of the parameter scan. Details of all the input parameters are discussed in the relevant section below.

Note that a 1D run is also possible! Just set scann=1 and then a k_θ spectrum for only one set of parameters will be calculated. A single wavenumber can also be run.

Once satisfied with the input file parameters, type in the command input_QLK_scan (from inside the MATLAB environment). This will create - in the run directory - a link to the QuaLiKiz executable, create a batch_$.pbs file for launching the run on the cluster, and create QuaLikiz input files inside the /input directory, based on the settings in the parameter.m file. You will also be prompted to type in the name of your job which will run on the cluster. For example, one can type 'run10' for the batch job name. Plots of the various input variables are displayed to visually examine some of the parameters for error checking.

At the moment, batch files are created for the following clusters: Andromede ($=a), Cephee ($=c), Gateway ($=g) and JAC ($=j).

To launch the run just type just type '!qsub batch_$.pbs' from within the MATLAB environment, or 'qsub batch_$.pbs' from the UNIX command line in your run directory. Note that $ denotes here the letter corresponding to the cluster you are on. It is often useful to launch the run interactively (i.e. to run on the front-end node). For an interactive run with one processor, type

'./QuaLiKiz.exe'. For an interactive run with parallel execution, type, e.g., 'mpiexec -np 8 QuaLikiz.exe', for 8 processors. Include an exclamation mark (!) before the above commands, if you are in a MATLAB environment. Note that the directory where the 'mpiexec' command is located, must be in the path in your UNIX environment.

3.2 Setting up a new CRONOS based run

For a CRONOS based run, the following steps need to be carried out to set up a QLK run from within the MATLAB environment.

1. A CRONOS file needs to be loaded in the background, i.e. the CRONOS data and param files. In the new run directory, the function getcrondat.m can be found. This is used to then save the relevant data for QLK directly from the CRONOS file. This function is called by typing 'getcrondat(data,param,nshot);' from within the MATLAB environment. nshot should be a shot number, used for bookeeping purposes. The user will be prompted to provide the time window desired for profile averaging, e.g. [45 45.5]. A single time slice can also be used by inputting the same time in both field, e.g. [45 45]. A matlab .mat data file is then saved in the run directory with the CRONOS data. The shot number appends the matfile name.

2. Make sure that the contents of the just made crondat#####.mat file is loaded in the MATLAB workspace. A crondat structure should then exist in the MATLAB workspace. Then, run the script 'make_CRONOS_parameterfile'. The user will be prompted for some basic info for the run (e.g. ktheta grid), and based on the crondat structure a parameters.m file will be made based on the CRONOS run parameters.

3. Run the script input_QLK_scan. This will then, as above, prepare the ground for the QLK simulation. !qsub batch_$.pbs will launch the run, or it can be launched interactively. Note that $ should be repalced by the letter corresponding to the cluster you are on. The parameters.m file can be modified by hand before each call to input_QLK_scan, in case input parameters are desired to be changed.

4 Detailed QuaLiKiz input

For now, the input parameters are written into binary files by a MATLAB script. The standalone version of QuaLiKiz then reads these binary files from the /input folder, and writes the output in the /output and output/primitive folders. For integration into the Transport Code Interface (TCI) for inclusion in JINTRAC and ETS, the inputs and outputs are passed through memory. This manual deals mainly with the standalone code.

4.1 Numerical parameters

phys_meth . If 0, outputs only total fluxes. If 1, outputs additional separate particle diffusivity and pinches. If 2, further outputs separate heat diffusivity and pinches.

coll_flag . If 0, collisionless simulation (slight speed-up due to 1D electron trapped integral). If 1, collisional simulation.

rot_flag . If 0, rotation not included in dispersion relation (but still included for heavy impurity asymmetries). If 1, rotation is fully included.

numsols . Maximum number of unstable solutions in output. Typically 3 is the maximum number of conceivable instabilities.

nprocs . Number of processors used in parallel computation. This is only used to set up the batch script.

maxruns . Typically used for integrated modelling. If maxruns ¿ 1, then this is the frequency for which solutions will be sought for in the entire defined range in the complex plane. For interceding QuaLiKiz calls, the solution will start from a Newton root finding using the previous set of solutions as the initial guesses. This saves lots of time in integrated modelling, but means that the previous solutions must be saved in memory or on disk.

maxpts . Maximum number of integrand evaluations done in the 2D integrals.

relacc1 . Relative accuracy in 1D integrals

relacc2 . Relative accuracy in 2D integrals. Major factor in setting runtime!

timeout . Time in seconds after which the code 'gives up' on a given solution. Important for reducing occasional difficult cases from being bottlenecks in integrated modelling. 30 is a recommended value, or at least a number that corresponds to around $\sim 20ms$ of physical time.

ntheta . Resolution of parallel grid in poloidal asymmetry calculation. Does not slow down the code, so a relatively high value of 64 is set as default.

set_ninorm1 . If 1, then the main (first) ion species density will be modified to maintain quasineutrality, based on the values of the other active ion species densities.

set_QN_grad . If 1, then quasineutrality of gradients will be checked in preprocessing.

set_Ani . If 1, then the main (first) ion species density gradient will be modified to maintain quasineutrality of gradients based on the values of the other active ion species density gradients.

ETGmult . Default = 1. A multiplier for the ETG saturation rule constant pre-factor. Mostly for testing purposes. No multi-scale model is available in QuaLiKiz

collmult . Default = 1. A multiplier for the collisionality. Mostly for testing purposes. QuaLiKiz has a crude Krook collisional operator, which demands detailed comparisons of TEM emergence at high collisionality with GK codes involving less approximations

4.2 Physical parameters

kthetarhos . Array of wavenumbers. $\rho_s = \sqrt{(}T_e m_i)/eB$

numn . Size of wavenumber array (automatically calculated from the input size of the kthetarhos array).

scann . Size of 'radial scan'. This is a scan of all the other physical parameters. In integrated modelling this is typically the radially dependent parameters. In standalone use, this could be scans of single parameters, e.g. magnetic shear (as in the parameter_template.m example file). Set by user.

4.2.1 Geometry

Magnetic geometry parameters.

Bo . Magnetic field at magnetic axis, in [T]. Used for GyroBohm normalization.

R0 . Major radius [m]. A scalar used in normalizations.

Ro . Major radius [m]. An array, which can include the Shafranov shift impact, for example

Rmin . Minor radius a [m]. Scalar.

x . Local radius in normalized minor radius, e.g. r/a. Used to set trapped electron fraction.

ρ . Normalized toroidal flux coordinate. More for convenience (debug outputs and comparisons with profiles). Not used in calculations.

qx . q-profile array

smag . Magnetic shear array. $s{\equiv}rq'/q$

alphax . MHD alpha array. Used in magnetic drift frequency. $\alpha_{MHD}{\equiv}q^2\sum_s\beta_s(R/L_{Ts}+R/L_{ns})$.

4.2.2 Electrons

Electron parameters.

Tex . Electron temperature [keV].

Nex . Electron density $[10^{19}m^{-3}]$

Ate . Normalized logarithmic electron temperature gradient. $Ate \equiv -\frac{R}{T_e}\frac{dT_e}{dr}$

Ane . Normalized logarithmic electron density gradient. $Ane \equiv -\frac{R}{n_e}\frac{dn_e}{dr}$

el_type . 1 for active electrons, 2 for adiabatic electrons, 3 for passing only at ion scales

anise . Electron temperature anisotropy. $T_\perp/T_\parallel$ at low-field-side (LFS). Tex above is defined as $T_\parallel$. The profile of the perpendicular temperature (and hence the temperature anisotropy) along the field line is set according to the mirror force. See Bilato NF 2014.

danisedr . Radial gradient of the temperature anistropy at the LFS.

4.2.3 Ions

Ion parameters.

Ai . Ion mass in amu

Zi . Ion charge in units of electron charge

Tix . Ion temperature [keV].

ninnorm . Normalized ion density n_i/n_e.

Ati . Normalized logarithmic ion temperature gradient. $Ati \equiv -\frac{R}{T_i}\frac{dT_i}{dr}$

Ani . Normalized logarithmic ion density gradient. $Ani \equiv -\frac{R}{n_i}\frac{dn_i}{dr}$

ion_type . 1 for active ions, 2 for adiabatic ions, 3 for tracer (i.e. for heavy impurities), and 4 for tracer but nevertheless to include the impact on Zeff and collisionality. Note that for 3, the ninorm value is arbitrary, but for 4 the ninorm value needs to be set consistently with quasineutrality.

anis . Ion temperature anisotropy. This is much more important than the electron anisotropy. $T_\perp/T_\parallel$ at low-field-side (LFS). Tix above is defined as $T_\parallel$. The profile of the perpindicular temperature (and hence the temperature anisotropy) along the field line is set according to the mirror force. See Bilato NF 2014.

danisdr . Radial gradient of the temperature anistropy at the LFS.

iind . Total number of ions in the calculation.

4.2.4 Rotation

All factors here are normalized by the reference velocity $c_{ref} \equiv \sqrt{T_{ref}/m_p}$, with $T_{ref} = 1keV$, and m_p the proton mass. All species are assumed to rotate with the same velocity (!). In principle, the toroidal, parallel, and perpendicular velocites can all be defined independently of each other. If there is a wish to constrain the values by assuming pure toroidal rotation, then the user (at the moment) must do this manually.

Machtor . Toroidal velocity normalized by c_{ref}.

Machpar . Parallel velocity normalized by c_{ref}.

Autor . Toroidal velocity gradient. Autor$\equiv -\frac{R}{c_{ref}}\frac{dv_{tor}}{dr}$.

Aupar . Parallel velocity gradient. Aupar$\equiv -\frac{R}{c_{ref}}\frac{dv_{\|}}{dr}$.

gammaE . Normalized perpendicular $E{\times}B$ flow shear. e.g., for pure toroidal rotation $\gamma_E{\equiv}\frac{\epsilon}{q}$Autor, with $\epsilon{\equiv}r/R$.

5 Detailed QuaLiKiz output

All the output is written in the /output directory, as ASCII files. /output/primitive contains more 'primitive' QuaLiKiz output (quasilinear integrations, modewidths, etc.) which are used by the saturation rule to generate the final output (e.g. fluxes) in the /output directory. These 'primitive' outputs, mostly used for consistency checks and debugging, are not detailed in this version of the manual.

At the moment there is no automated plotting tool to view the results. However, since it's ASCII, it's very easy to write your own plotting scripts.

5.1 Main output

Note that some of the output is optional (e.g. the separation of diffusivities and pinches) depending on choices in the numerical parameter input.

gam_GB are the growth rates in gyrobohm units (normalized to $\sqrt{Te/mi}/a$)

gam_SI are the growth rates in angular frequency (s^-1)

ome_GB are the frequencies in gyrobohm units. Negative values are in the ion diamagnetic direction (e.g. ITG), positive are in the electron direction (e.g. TEM).

ome_SI are the frequencies in angular frequency (s^-1)

The above growth rate and frequency arrays are organized as follows: the columns correspond to wavenumbers, the rows to the scan number (typically corresponding to radius). This repeats itself for the total number of solutions saught for (default numsols=3).

ief_GB is the ion heat conductivity, χ_i, in gyrobohm units: (normalized to $\sqrt(mi)T_e^{1.5}/(q_e^2B^2a))$

ief_SI is the ion heat flux in SI units (W/m^2)

eef_GB is the electron heat conductivity, χ_e, in gyrobohm units: (normalized to $\sqrt(mi)T_e^{1.5}/(q_e^2B^2a))$

eef_SI is the electron heat flux in SI units (W/m^2)

eefETG_SI is the electron heat flux from electron scales only, in SI units (W/m^2)

ipf_GB is the total ion particle diffusivity in gyrobohm units (assuming only diagonal transport)

ipf_SI is the total ion particle flux in SI

epf_GB is the total electron particle diffusivity in gyrobohm units (assuming only diagonal transport)

epf_SI is the total electron particle flux in SI

epfETG_SI is the total electron particle flux from electron scales only, in SI

ivf_GB is the total ion momentum diffusivity in gyrobohm units

ivf_SI is the total ion momentum flux in SI

evf_GB is the total electron momentum diffusivity in gyrobohm units (very small, but here for completeness)

evf_SI is the total electron momentum flux in SI

dfe_SI electron diffusivity (the term proportional to dn/dr) in SI units

vte_SI the electron particle thermopinch in SI units

vri_SI the electron particle rotodiffusion pinch in SI units

vce_SI the electron particle compressibility pinch in SI units

dfi_SI ion diffusivity (the term proportional to dn/dr) in SI units

vti_SI the ion particle thermopinch in SI units

vri_SI the ion particle rotodiffusion pinch in SI units

vci_SI the ion particle compressibility pinch in SI units

chiee_SI electron heat conductivity (the term proportional to dTe/dr) in SI units

vene_SI the electron heat thermopinch in SI units

vere_SI the electron heat rotodiffusion pinch in SI units

vece_SI the electron heat compressibility pinch in SI units

chiei_SI ion heat conductivity (the term proportional to dTi/dr) in SI units

veni_SI the ion heat thermopinch in SI units

veri_SI the ion heat rotodiffusion pinch in SI units

veci_SI the ion heat compressibility pinch in SI units

cke Consistency check on electron particle transport. 100*(epf - (-dfe*dne/dr + (vce+vte)*ne))/epf. Should be on the order of relacc2*100.

cki Consistency check on ion particle transport. 100*(ipf - (-dfi*dni/dr + (vci+vti)*ni))/ipf. Should be on the order of relacc2*100.

ceke Consistency check on electron energy transport. 100*(eef - (-chiee*dTe/dr + (vece+vene+vere)*Te))/eef. Should be on the order of relacc2*100.

ceki Consistency check on ion energy transport. 100*(ief - (-chiei*Ti/dr + (veci+veni+veri)*Ti))/ief. Should be on the order of relacc2*100.

eef_cm . Wavenumber dependent electron heat flux spectrum (SI units). Columns are different wavenumber, rows different 'radial coordinates'

epf_cm . Wavenumber dependent electron particle flux spectrum (SI units). Columns are different wavenumber, rows different 'radial coordinates'

evf_cm . Wavenumber dependent electron momentum flux spectrum (SI units). Columns are different wavenumber, rows different 'radial coordinates'

ief_cm . Wavenumber dependent ion heat flux spectrum (SI units). Columns are different wavenumber, rows different 'radial coordinates', and this is repeated for every ion.

ipf_cm . Wavenumber dependent ion particle flux spectrum (SI units). Columns are different wavenumber, rows different 'radial coordinates', and this is repeated for every ion.

ivf_cm . Wavenumber dependent ion momentum flux spectrum (SI units). Columns are different wavenumber, rows different 'radial coordinates' and this is repeated for every ion.

modeflag . If 0, both electron and ion modes important. If 1, only ion modes important. If 2, only electron modes important. 'Radial' array.

Note that all the ion flux arrays above are organized as follows: the columns correspond to the different ions, the rows to the scan number (typically radius). This is different for the 'cm' arrays (flux spectra), as specified.

5.1.1 Poloidal asymmetry related output

For the e-coefficients defined below, we use the following auxilliary definitions: $A{\equiv}T_\perp/T_\parallel$, anisotropy at the LFS. $P{\equiv}\frac{1}{A-(A-1)\frac{1+\epsilon cos\theta}{1+\epsilon}}$, poloidally dependent anisotropy assuming small inverse aspect ratio (consistent wtih QuaLiKiz geometry). $E{\equiv}Ze\Phi(r,\theta)-\frac{m\Omega^2(r)}{2}(R^2(\theta,r)-R_{LFS}^2)$, the energy including rotational kinetic and electrostatic energy. Species dependent. Φ is the electrostatic potential as calculated by quasineutrality along the field lie. Ω the angular velocity. The $\langle\cdot\rangle$ operator denotes a flux surface average. M is the Mach number normalized by the ion sound speed.

npol . Ion density poloidal profiles (due to poloidal asymmetries). Radial coordinate varies by column, poloidal by row, and this is repeated for every ion. Thus, e.g., for 3 ions and 64 poloidal gridpoints, this output array will have 192 rows.

ecoefs . Poloidal asymmetry 'e-coefficients' calculated for benchmarking with GKW and NEO and for calculating the cftrans coefficients. There are 13 such coefficients. This array contains 13 columns (1 per coefficient), and nion+1 rows. The 1st row corresponds to electrons, and the next rows to the ions. This is repeated for each radial point.

ecoef0 $= \langle Pe^{-E/T}\rangle$

ecoef1 $= \langle \frac{Ze}{T}\Phi Pe^{-E/T}\rangle$

ecoef2 $= a\langle \frac{Ze}{T}\Phi' Pe^{-E/T}\rangle$

ecoef3 $= \frac{1}{a^2}\langle (R^2(\theta)-R_{LFS}^2)Pe^{-E/T}\rangle$

ecoef4 $= \frac{2}{a}\langle (R(\theta)R'(\theta)-R_{LFS}R'_{LFS})Pe^{-E/T}\rangle$

ecoef5 $= 0$ (in Hamada coordinates)

ecoef6 $= a\langle P'e^{-E/T}\rangle$

ecoef7 $= \langle Pe^{-E/T}\frac{s}{\epsilon}\theta\frac{\partial\Phi}{\partial\theta}\rangle$

ecoef8 $=\langle Pe^{-E/T}(1+\epsilon cos\theta)(cos\theta+s\theta sin\theta)\rangle$

ecoef9 $= \frac{2}{R}\langle -R_{LFS}\frac{dR_{LFS}}{dr}Pe^{-E/T}\rangle$

ecoef10 $= R\cdot\langle dn/dr\rangle/\langle n\rangle$

ecoef11 $= \langle n\rangle$

ecoef12 = Asymmetry factor $(n_{LFS}-n_{HFS})/\langle n\rangle$

cftrans . Ion transport coefficients following post-processing due to poloidal asymmetries. Imporant for heavy impurity transport. There are 6 columns corresponding to the different coefficients as defined below. The rows correspond to radial coordinates. This is then repeated for every ion. e.g., for 5 ions and 16 radial gridpoints, there will be 80 rows. Column 1 is generalized diffusivity . Column 2 is generalized thermopinch. Column 3 is generalized compression pinch. Column 4 is the "2D thermopinch".

Column 5 the "2D rotodiffusion pinch" . Column 6 is the "2D pure pinch". 'Generalized' means here that the 2D poloidal density weighting is taken into account (i.e. with ecoef0). '2D' means that the all the geometric asymmetry effects are taken into account. Specifically:

cftrans1 = $D\cdot$ecoef0

cftrans2 = $V_t\cdot$ecoef0

cftrans3 = $V_c\cdot$ecoef0

cftrans4 = $\frac{1}{R}A_{ti}D(\text{ecoef1} - \text{ecoef3}\frac{m(\Omega_{tor}R)^2}{2T})$

cftrans5 = $\frac{2}{R}D\cdot M\cdot\text{Aupar}\cdot(c_{ref}/c_{th})\sqrt{1+(\epsilon/q)^2}\text{ecoef3}$

cftrans6 = $\frac{1}{R}D(\text{ecoef2} - \text{ecoef7} - 2M^2(\text{ecoef8} + \text{ecoef9}))$

Where D is the ion particle diffusivity as calculated without poloidal asymmmetries.

6 General notes

When running the code, do not be alarmed if there are some errors reported in the output stream regarding integration convergence. QuaLiKiz carries out thousands of such integrations during the computation of each eigenvalue and quasilinear flux. Some do not converge. However, generally, this does not significantly the final result. However, this behaviour can cause some fluctuations in the fluxes and profiles in integrated modelling calculations. This makes it difficult for fully implicit solvers to converge. Also, some convergence failures can appear occaionally in the fluid eigenfunction solver. The code then reverts to a less self-consistent fluid solution. Again, this typically does not lead to a significant perturbation of the final results.

When running on a restricted number of points in a stand-alone fashion, it may be advantageous to carry out convergence tests on the results by increasing the integration accuracy inputs. This is relacc1 and relacc2 in the input parameter file. Results are significantly slower with higher relacc1 and relacc2 than the default values. For integrated modelling, it is highly recommend to maintain the default values, which were chosen after extensive testing to optimize the balance between speed and accuracy.

It is recommended to run the code in parallel. For example, one Andromede node has 12 processors, so this is a recommended number to use on Andromede. An interactive run can be carried out with 'mpirun -np 12 QuaLiKiz.exe'. This is a very convenient way to run the code for standalone applications, especially if queue waits are involved in batch mode.

Hopefully, this manual is sufficient to get started. Clearly the manual can be much further improved. For more detailed questions and suggestions please write us.

Finally, note that we are now porting the standalone interface from MATLAB to Python. Therefore, no further development is planned on the MATLAB

I/O scripts. Unfortunately we have no generic output plotting script available. We recommend writing your own scripts for your own specific use cases.

Happy computing!