# An Outline of Onetwo Code Structure

H.ST.JOHN

## 1 Brief Description

The Onetwo transport code solves a user selectable,arbitrary combination,of flux surface average parabolic equations representing the radial diffusion and convection of 1 or 2 thermal ion species and up to 2 neutral species,the electron energy,the ion energy,the poloidal magnetic field,and the toroidal momentum. The magnetic geometry may be self consistently maintained by solving the elliptic mhd equilibrium equation simultaneously with the diffusion equations. Both free and fixed plasma boundary equilibrium models are supported. Auxiliary heating models include Monte Carlo neutral beam deposition and electron and ion cyclotron and fast wave r.f. heating. Models for ohmic,bootstrap, beam and r.f. driven currents are included. Energy confinement may be simulated by selecting theoretical or empirical electron and/or ion thermal conductivity models,including Rebut-Lallia,Shay,Waltz-Dominguez and IFS. A unique feature of the code is its ability to substitute measured profiles for any of the dependent variables and solving the diffusion equations only for the remaining unknown profiles. A combination of Crank-Nicholson,predictor and iterated corrector methods are used to converge the non linear diffusion equations. The elliptic equations are solved using variable finite difference schemes with single cyclic reduction. The code is strictly procedural in nature,written entirely in F77 (with the "standard" extensions) . *ONETWO* consist of over 300 subroutines. Only the most significant ones are listed below. The code consists of a set of functionally grouped subroutine modules as follows.

| | |
|---|---|
| cray001.f | date/time-stamp and block data routines |
| cray101.f | main driver routines |
| cray102.f | input routines |
| cray201.f | mhd routines |
| cray202.f | support routnes for cray201.f |
| cray203.f | more support routines for cray201.f |
| cray204.f | iter data base routines (netcdf and ascii) |
| cray205.f | netcdf routines for tdem mode |

| | |
|---|---|
| cray206.f | tdem mode routines |
| cray301.f | main transport driver routines |
| cray311.f | neutral transport routines |
| cray321.f | beam routines |
| cray331.f | rf related routines |
| cray341.f | pellet injection related routines |
| cray352.f | rf routines |
| cray401.f | general support routines |
| cray402.f | Neutron rate calculation routines |
| cray403.f | Gaussian integration routines |
| cray404.f | Nordman-Weiland confinement model |
| cray501.f | adaptive grid routines(under development) |

# 2 Input/Output

*ONETWO* uses namelist input for all of its kinetic,beam,and rf data input. Magnetic geometry related input is done through special ascii files called eqdsks or through netcdf files (the TDEM mode,see below).Profile data input is in the form of time dependent splines or parameterized parabolic shapes. The kinetic profiles(densities and temperatures) and the current profile (no option for q at present) cn be input at pre-specified times if the corresponding diffusion equation is run in analysis mode. Output is to ascii and netcdf files. Graphics output is done with display and/or IDL,both require special post-processing programs (trplot and plot12 respectively).

**Init(cray102.f)** the namelist input and code initialization routine. Also contains a detailed description of all input variables.

prepar writes data intended for graphics visualization (both dissplay and IDl type)

out primary output routine,writes to ascii file outone

preplt time orders data from prepar, creates netcdf output file

**trplot** creates dissplay style graphics file

**plot12** creates interactive,netcdf driven,IDL graphics.

# 3 MHD

The *ONETWO* code is structured in a manner which allows mhd calculations to proceed with a relatively long time step compared to the time steps taken in the transport calculations. This is accomplished by wrapping the transport calculations up inside an mhd module (runtwo)which calls the transport module (tport). The time steps between mhd calculations are input by the user (as a heterogeneous list of times at which the equilibrium problem is to be solved). This list of times is driven by the fact that linear interpolation is used to connect the mhd calculations from one time step to the next. Hence the user has the ability to allow long time periods of uninterupted transport evolvement when appropriate by picking large time intervals between mhd calculations. On the other hand, if rapid plasma shaping is occurring, the time steps can be made short as appropriate. The code will accept the input time steps or repeatedly cut the given input time steps in half if the geometric parameters associated with the mhd calculations (ie the metrics)are not converged. That is, within any given mhd cycle there is a transport cycle which evolves the transport equations from the begging to the end of the mhd time step. The time dependent mhd derived quantities required to do the transport calculations are linearly interpolated. The slope of the interpolation function is intially assumed zero (at program start up) and is taken as the continuation of the slope from the previous mhd cycle thereafter as a starting guess. This guess may have to be refined if it is too far off the value that the current mhd calculation gives. If an iteration of the mhd/transport cycle is required then a new slope is determined by combining the previous and current calculations (ie under relaxation) of the mhd metrics to form a new slope for the interoplation function ( there is separate time slope function for each sptial grid point). There a good and bad points to this approach. The bad point is that an entire transport cycle set of calculations (which could consist of a large number of transport time steps) may be rejected if the mhd metrics are too far off at the end of the cycle. One good point is that there is a lot of flexibility associated with this approach. A common way of running the code is to force a single, time independent mhd cycle ( which can easily be set up with the above approach).

The code does not have external circuit equations. Instead it uses the (experimentally measured) values of the psi loops and magnetic probes (as a function of time)to determine what the plasma boundary and flux surface shape should be.

Because equilibrium/transport cycles are expensive and because of the way that experimental data is analyzed there is an option in *ONETWO* to use time dependent equilibrium calcualtions which were done a priori. In an experimentally driven program it is typicall that many stand alone equilibirum calcualtions are performed (using an equiibrium code such as efit) as function of time to fit measured data to the magnetic geometry required for data analysis. Thus many equilibrium files (called eqdsks) are available before any transport calcualtions are attempted. In the TDEM mode of operation (time dependent eqdsk mode) the Onetwo code is instructed to forget about doing equilibrium calculations altogether and instead it uses the available eqdsks to construct the required information as a function of time as the transport calcualtions proceed. The previously generated set of eqdsks is processed by a stand alone code,mepc (multiple eqdsk processor) which writes a netcdf file that is actually used by *ONETWO* .

**runtwo(cray201.f)** main driver for mhd/transport cycle

**drive** driver for mhd calculations

**solvegs** sets up finite diff matrices for equilibirum equation

**cmpltcyr** (single) cyclic reduction solver

**bound** free boundary finder

**cntour** internal flux surface generator

**fluxav** does all flux surface avg calculations

**sorpicard** sets up fixed boundary calculatioins for Picard iteratioin solution.

**prepar** determines input for transport section

**iterdb** write iter database file (netcdf or ascii)

# 4 Transport

The transport calculations are structured to use a Crank-Nicholson type approach for the space and time differencing of the diffusion equations. The nonlinearities are resolved by using a predictor corrector method. A time step starts by calculating the sources (subroutine source) which may include neutral beam and rf heating. The rf packages (toray,curray) are stand alone codes which are spawned form *ONETWO* and return results to *ONETWO* through ascii files. The frequency of the calls to the beam and rf packages is determined by the user. The evolution of the poloidal magnetic field and the associated parrallel and toroidal currents with auxilliary current drive is probably adequately explained in the document

`(on hydra)/u/stjohn/ONETWOPAGES/code_descr/sources_o12.ps`

. Otherwise the solution of the density and (electron,ion) energy equations is standard (all ions are assumed to have a single common temperature). Modelling of impurities is limited to simple charge balance calculations . The relevant subroutines are

**tport(Cray301.f)** the driver routine. Controls the time stepping and convergence estimates, makes all calls to determine transport coefficients, sources, mhd parameters.

**rhomsh** generate the rho mesh (square root of toroidal flux) at current time (alows for expanding boundary and adaptive grid calculations).

**diffus** sets up diffusion coefficient matrix

**fluxx** determines the flux forboth analysis and simulation mode

**zen** determines ion densities consistent with fast beam ions and fusion alphas.

**specify** sets values for analysis mode variables to current time

**source** determines all sources,including neutral densities

**neucg** neutrals code,based on 1d circular cylinder approximation the neutral density at any instant is determined by the reaction rates required to sustain the plasma with the current(te,ti) temperatues and densities (primary ions, impurity ions,and fast ions). This section of the code needs updating for more complete reaction rates, cross sections and 3d geometry.

**solve** decomposes and solves the set of nonlinear matrix equations representing the transport system of equations.

**cheku** checks for valid solution. Sets flag to cut time step in half if valid solution is not found.

**checksolution** determines if a corrector step is required

**ballo** does balloning mode calcs

**propel** does pellet injection related calcs

**fiziks** determines physical quantities(eq relaxation times,etc) from solution at current time step

**freya** does neutral beam depostion

**orbit** does prompt fast ion orbit averaging

**ech** gives ech power depostion based on simple internal models or spwans exteranl rf codes

**tspline** primary tension spline routine

**hirsh88** Hirshman bootstrap current model