NEWTON
Multiphysics coupling master code
User's manual

Federico Caccia

June 27, 2017

Contents

Preface

Newton is a master code that solves explicit and implicit coupling in nonlinear calculations. It was designed with a general purpose. For example, it can be used in multiscale coupled problems, in CFD-temalhydraulic problems, in neutronic-termalhydraulic problems, and others. These are the main features of the code:

- **Capable of coupling all kind of codes.**
- **Easy to use.**
- **Respect the GNU project philosophy.**

This is the user's manual. Here you find everything necessary to prepare the input and solve a coupling problem.

1

Input cards

1.1 General cards

1.1.1 card METHOD

Sets the nonlinear method used to solve the residual of the equations. Available values are:

```
METHOD  explicit_serial
```

This method is the common dirichlet-to-newmann explicit method. Codes run by phases inside each iteration. Initial guesses are sended to clients in phase run. Newton waits for their solutions and after that, updates unknowns and send necessary values to client codes in phase 2, and so on.

```
METHOD  explicit_parallel
```

This method is similar to the *explicit_serial*, but in this case, guesses are sended to all client codes at the same time. Once all client has ended their calculations, Newton updates the unknown solution with these values and send them again to the clients codes as new guesses.

```
METHOD  newton
```

Newton method. It builds the Jacobian matrix in each iteration, so it requires at least $N+1$ function evaluations in each iteration (with N amount off coupled client codes).

```
METHOD  secant
```

This method builds the Jacobian matrix only at the beggining of the iterations. It is possible to update the matrix until a determined amount of iterations or steps in evolution problems using ITER_JAC_CALC and STEPS_JAC_CALC cards.

```
METHOD  broyden
```

Broyden method is a quasi-newton method with superlinear convergence order. It is possible to initialize the Jacobian matrix with a guess by J_INI card or with calculation of the Jacobian matrix by finite difference (Otherwise Newton starts using the identity matrix). Also, it is possible to recalculate the matrix until a determined amount of iterations or steps in evolution problems using ITER_JAC_CALC and STEPS_JAC_CALC cards.

1.1.2 card PHASES

This card is only necessary using explicit_serial method. Client code names should be provided in the same order that it is desirable to run. Phases are separated using "&". To end the enumeration, use "&" too. For example, in a problem in wich *client*1 and *client*2 should run in phase 1, *client*3, *client*4 and *client*5 should run in phase 2 and *client*6 should run in phase 3:

```
PHASES  client1 client2 & client3 client4 client5 & client6 &
```

It is also posile to use inner iterations in each phase. To hablitite this option, use PHASES_MAX_ITER card.

1.1.3 card PHASES_MAX_ITER

This option can be only used in EXPLICIT_SERIAL method. It allows to use inner iterations in each phase. To use it set amount of maximum iterations for each phase. For example, if we desire to use up to 10 iterations in phase 1, and just 1 in phase 2, set:

```
PHASES_MAX_ITER  10 1
```

or just

```
PHASES_MAX_ITER  10
```

to leave phase 2 with defafult optiones(just 1 inner iteration).

1.1.4 card ABS_TOL

Absolute nonlinear tolerance in nonlinear iteations to reach the convergence of the solution. Newton step ends with WARNING when norm 2 of the residual falls below ABS_TOL.

```
ABS_TOL  1e-14
```

1.1.5 card MAX_ITER

Maximum amount of nonlinear iteations allowed to reach the convergence of the solution. Newton step ends with WARNING when nonlinear iterations grows above MAX_ITER.

```
MAX_ITER  100
```

1.1.6 card X_EXT_ORDER

Order of extrapolation to set guess at new evolution step. Now it is only availale order 1. Use:

```
X_EXT_ORDER  1
```

1.1.7 card J_EXT_ORDER

Order of extrapolation to set jacobian at new evolution step. It can be used with any implicit method. Now it is only availale order 1. Use:

```
J_EXT_ORDER  J
```

1.1.8 card X_INI

Initial condition in unknowns. It can be used as general card or inside CLIENT card. After this card set unknown name and unknown value. For example, to set $x_0 = 0.1$ and $y_0 = 0.2$ values, use:

```
X_INI  x 0.1 y 0.2
```

1.1.9 card STEPS_JAC_CALC

It can be used with any method that builds system's jacobian, but not for newton (newton computes jacobian in each step and iteration). It sets difference between steps in wich jacobian is computed by finite difference. Its default value is 0. To change it to 100 for example, use:

```
STEPS_JAC_CALC  100
```

1.1.10 card ITER_JAC_CALC

It can be used with any method that builds system's jacobian, but not for newton (newton computes jacobian in each step and iteration). It sets difference between iterations in wich jacobian is computed by finite difference. Its default value is 0. To change it to 100 for example, use:

```
ITER_JAC_CALC  100
```

1.1.11 card DX_JAC_CALC

It can be used with any method that builds system's jacobian. It sets delta in unknown to compute residual derivate by finite difference. Its default value is 0.1. To change it to 0.01 for example, use:

```
DX_JAC_CALC  0.01
```

1.1.12 card N_STEPS

Number of evolution coupling steps. For example, in a transitory problem with 10 coupling time steps use:

```
N_STEPS  10
```

1.1.13 card DELTA_STEP

Difference in evolution parameter between two steps. Its units depend on the problem. For example, in a neutronic neutronic-termalhydraulic coupling problem solving quasi-static steps in a burnup evolution, if we update cross sections every 50 dayys, use:

```
DELTA_STEP  50.0
```

1.1.14 card CLIENT

This is a block card. After setting this card, all CLIENT options can be set, until a new general card or end of file is found. The first option that has to be set is client name. To set *client*1 client options use:

```
CLIENT  client1
...  ...
```

Use any of the CLIENT cards after that.

1.1.15 card MAPPER

This is a block card. After setting this card, all MAPPER options can be set, until a new general card or end of file is found. The first option that has to be set is mapper name. To set *map*1 mapper options use:

```
MAPPER  map1
...  ...
```

Use any of the MAPPER cards after that.

1.1.16 card DEBUG_TIME

It can be used to export time calculation values in *time.log*. To use it just set:

```
DEBUG_TIME
```

1.2 Client cards

Inside CLIENT block (see general card CLIENT), it can be used any of these cards:

1.2.1 card CONNECTION

This is an obligatory card. Select one of the connection mode options:

```
CONNECTION  io_spawn
```

This option is used to communicate master with client by input / output option spawning N processes of the client by $MPI_Commspawn$. $MPI_Commspawn$ doesn't wait that the slave ends running and so it is needed an MPI_Barrier after the spawn. Client should have implemented this barrier too, once the output has been printed. $newton_spawn$ needs also theese cards: N_PROCS, INPUT_NAME, INPUT_EXT, OUTPUT_NAME, OUTPUT_EXT, BIN_COMMAND, ARGS and IO_TYPE.

```
CONNECTION  io_system
```

This option is used to communicate master with client by input / output option spawning 1 process of the client by *system* function of the standard library of c++. *system* waits that the slave ends running and so it isn't needed any barrier as in $newton_spawn$. NEWTON_SPAWN needs also theese cards: N_PROCS, INPUT_NAME, INPUT_EXT, OUTPUT_NAME, OUTPUT_EXT, BIN_COMMAND, ARGS and IO_TYPE.

```
CONNECTION  mpi_port
```

This option is used to communicate master with client by mpi functions. Master publishes a port to connect with client and then connection has to be stablished using some functions in client. Also, some functions related to send and receive variable values have to be implemented in client. See ?? section to understand how to implement this communication mode in client.

```
CONNECTION  mpi-comm
```

This option is used to communicate master with client by mpi functions. Newton and client codes should be run from beggining using *mpirun*. Also, some functions related to send and receive variable values have to be implemented in client. See ?? section to understand how to implement this communication mode in client.

1.2.2 card IO_TYPE

This card is obligatory using io CONNECTION option. If input and output client files are simple (just variable values writen inside), it can be used:

```
IO_TYPE  test
```

If input and output client files are complex (not just variable values writen inside), it needs some programmed lines in *userClient.cpp* to help Newton to read and write these files. Use:

```
IO_TYPE  USER_CODE
```

Other options pre-programmed are:

```
IO_TYPE  RELAP_POW2TH
```

This mode is used running *RELAP* client with power fraction distribution as input. From the output there are extracted fuel temperatures, refrigerent temperatures and densities in the core.

```
IO_TYPE  FERMI_XS2POW
```

This mode is used running *FERMI* client with cross sections as input. From the output there are extracted power spatial distribution values.

```
IO_TYPE  NEUTRONIC_CR2KP
```

This mode is used running *FERMI* client with control rod positions as input. From the output there are extracted power spatial distribution values and k effective of the core.

```
IO_TYPE  NEUTRONIC_KP2CR
```

This mode is used running *CR* client with power spatial distribution values and k effective of the core as input. From the output there are extracted control rod position values.