Requirements for "GOLD" in Model Database

REQUIREMENTS FOR "GOLD" SUPPORT IN MODEL DATABASE

Background

When getting the model (Rmats or Twiss) for a given device through AIDA, the model server goes through a multi-step search to establish from which model run to return data.

The present behavior is that: if the RUNID parameter is given, the model is taken from the specified run. Otherwise, if the MODE parameter is given, the model returned is the chronologically most recent of the specified mode; if the MODE parameter is not given, the model returned is the chronologically most recent mode 5 model of the device if it is mode 5, and simply the most recent chronologically uploaded model of any mode otherwise.

The algorithm above is given in summary - it's more involved when the Rmat from A to B is requested. The algorithm for choosing the run from which to get the model is summarized in [1], and described at length in [2].

Problem

Since it's largely the last model to be uploaded, models can't be uploaded without affecting running physics software and operations, and therefore it's difficult to experientially test a model, or temporarily upload one for a given experiment, Energy profile, etc.

Also, the chronologically most recent model of each mode isn't completely obvious to the eye in the model web page (nor possibly in the coming Optics application), and the search algorithm is not completely obvious.

Solution

We will support the designation of a single "gold" model of each model mode (effectively a beamline). Other things being equal then, requests for model data of devices in a given mode will be taken from the gold model for that model mode. You would be able to change which is the gold model through the APEX web page [3], and through Paul and Quan's new optics GUI.

No changes are required on the client side (no matlab changes), though users should be aware of the change in behaviour.

See enclosed example screens:

- Run Data table showing new GOLD column [4].
- Gold History [5].

GOLD and MODE

Each MODE can be assigned (at most) one EXTANT GOLD model, and one DESIGN GOLD model, see picture.

> NOTE: So golds will not go with beamline per se. Choosing to associate gold with mode is the choice which makes most sense if mode is primarily used to identify such things as different initial conditions or ending energy.

GOLD and DESIGN models

We will allow DESIGN models to be designated "gold" too. This is a consequence of Paul Chu's plan to run and upload an energy scaled design model every time an energy scaled extant model is uploaded. If there were no GOLD design model in that case, every time the energy scaled extant model was run and uploaded, requests for the design model would get different data to before the model run. It's more intuitive if design models for a mode are pretty constant.

Note that, if an EXTANT model is made GOLD, it would NOT necessarily be true that any DESIGN model for taht same mode computed and uploaded at the same time would become the design GOLD of that mode too. We would leave the designation of which DESIGN is teh gold design entirely up to users, just as for extant models.

It's possible for a given mode not to have any model runs designated gold.

FUNCTIONAL REQUIREMENTS

This section lists functional requirement changes by subsystem. There are 4 subsystems involved:

1) AIDA (to change which model from which it gets data)

2) Oracle (to support GOLD in model subschema, and modifications to oracle procedures)

3) Model upload page (to support setting which model is GOLD, and history)

4) Model Optics GUI (to support setting which model is GOLD, and history).

AIDA

The AIDA interface for model requests will be modified to add 2 more permissible values for the RUNID parameter (0 and 1).

RUNID = 0 - means get from the latest model
RUNID = 1 - means get from the gold model
RUNID = nnn - means get from the given numbered model (as now)

RUNID = 0 will be the default. This will preserve the behavior of existing clients, since presently if RUNID is not given, the latest model is used.

device [RUNID=0|1|nn] [MODE=mm] [TYPE=DESIGN|EXTANT] [POS=BEG|MID|END]

AIDA's model search algorithm

(In all of the search criteria below, it's assumed that the type, "Design" or "Extant", is also a conjunction in the search criteria).

The combinations of RUNID and MODE for which the resulting behavior must be specified:

1) runid = nn
   [mode is ignored]

2) runid = 1 (gold)
   mode given
   mode not given

3) runid = 0 (latest) - the default
   mode given
   mode not given

The behavior under each combination above, for both cases of just getting the model of one device (A), or Rmat from A to B, is given below:

a) If only a single device's model (A) is required:

1) If a run ID is given (RUNID>1), then look for the device (A) in the model identified by the given Run ID. If A is not found in the given Run ID model, return an error "device not found in run id."
   [This is unchanged from before GOLD was added]

2) If Run ID = 1 (GOLD) is given;

   2.1) If MODE is given, then return the model parameters from the gold model of the given mode. If A is not in the gold model of the given mode, return an error (do not search further).

   2.2) If MODE is not given, then;

      2.2.1) If A is in the MODE = 5 gold model, then return the params of A from that model.

      2.2.2) If A is NOT in the MODE = 5 gold model, then return params of A from the gold model with the largest Run ID (that is, chronologically latest uploaded) that contains A.

3) If Run ID = 0 (LATEST) - the default;

   3.1) If MODE is given, then return the model params of A from the
        model with the largest run ID of the given MODE that contains
        A. If A is not any model matching the given MODE, return an
        error like "device not found in MODE".

   3.2) If MODE is NOT given, then;

        3.2.1) If A is in the MODE = 5 model, then return the params
               of A from model largest Run ID whose MODE == 5.

        3.2.2) If A is NOT in a MODE = 5 model, then return the params
               of A from the model with the largest Run-ID containing
               A. This is the present behavior.

b) If the model Rmat A to B is required

(In all of the search criteria below, it's assumed that the type,
"Design" or "Extant", is also a conjunction in the search criteria).

1) If a (>1) run ID is given, then look for A and B in the model
   identified by that Run ID. If A or B is not found in the model with
   the given Run ID, return an error like "device not found in run
   id."

2) If Run ID = 1 (GOLD) is given;

   2.1) If MODE is given, then return the model parameters computed
        from A and B Rmats from the gold model of the given mode. If A
        or B is not in the gold model of the given mode, return an
        error (do not search further).

   2.2) If MODE is not given,

        2.2.1) If A & B are in the gold MODE = 5 model, then return
               the params computed from A and B from that model.

        2.2.2) If A or B is not in the gold MODE 5 model, then return
               the params computed form the gold model with largest run
               id (chronologically most recent) that contains both A
               and B.

3) If Run ID = 0 (LATEST) - the default if RUNID param is not specified

   3.1) If MODE is given, then return the model params of A and B from
        the model with the largest run-ID that matches the given MODE,
        and contains both A and B. If A or B is not in the model with
        the largest run ID matching the given MODE, return an error
        like "device not found in MODE."

   3.2) MODE is NOT given, then

        3.2.1) If A & B are in any MODE = 5 model, then return the
               Rmat computed from A and B of the model with the
               largest Run ID, whose MODE == 5, that contains both
               (see note a.).

               Note a) For expediency of coding, I'm going to weaken
               this requirement to:

               3.2.1) Attempt to return Rmats computed from A and B
                      from the model with the largest Run ID, whose
                      MODE == 5, that contains A and B. However, if
                      the largest Run-ID model of MODE = 5 that
                      contains the downstream device, does not in
                      fact also contain the upstream device (which of
                      course it should), then return an error.

        3.2.2) If A or B is NOT in a MODE = 5 model, then return the
               Rmat computed from A and B from the model with the
               largest Run ID that contains both A and B. This is the
               present behaviour (see note b).

               Note b) For expediency of coding, I'm going to weaken
               this requirement to:

               3.2.2) If A or B is NOT in a MODE = 5 model, then
                      return the Rmat computed from A and B from the
                      model with the largest Run ID containing the
                      downstream device. This is the present behavior
                      (see note b).

ORACLE
======

The above functional requirements in the AIDA model data
provider, may be completed with the following addition to the
Oracle MACHINE_MODES sub-schema, and oracle code.

Machine_model sub-schema

Add 1 small table, Gold, to Machine_model:

Gold

ID (PK)
runs_ID (FK) [The foreign key to Runs table, ID]
Date   [The date the runs_ID runid above was made gold]
Comment   [User entered comment when setting this gold]
Created_by   [The user name of the user who made this model gold]

Oracle Procedures

Below are the required modifications to Oracle code:

1) GET_MODE_RUNID

Add Gold arg, as 4th arg, taking values 1 or null:

    1 = The run found must be a gold run (the present Gold of the
    given mode).

    null = Run need not be gold

If mode is not given, and Gold = 1, then return the run ID of the model
with the largest run ID which is gold (per a & b 2.2.2 above):

Modification to Aida:
{code}
                        "Select MACHINE_MODEL.ONLINE_MODEL_PKG.GET_MODE_RUNID("+
                        "'" + instance +"'," +
                        (type.equals("NULL") ? "null" : "'"+type+"'") + "," +
                        (mode.equals("NULL") ? "null" : "'"+mode+"'") +
        add line ->>   (run == 1 ? 1 : "null") +
                        ") as run_id from dual";
{code}

2) GET_MODE_TWISS_DATA

Expand permitted values of 5th param Run-ID, so that:
       null = latest (as now)
       1 = gold
       > 1 = match to argument (as now)

Curiously, run id param of GET_MODE_TWISS_DATA seems to be char, not
numerical? Why not NUMBER?

Aida code change to users of GET_MODE_TWISS_DATA:
get_twiss: - no change necessary, unless we decide to change run-Id arg to NUMBER.

{code}
                    "SELECT KINETIC_ENERGY, " +
                        "PSI_X, BETA_X, ALPHA_X, ETA_X, ETAP_X,"+
                        "PSI_Y, BETA_Y, ALPHA_Y, ETA_Y, ETAP_Y,"+
                        "ZPOS, LEFF, SLEFF, ORDINAL FROM "+
                        "TABLE(MACHINE_MODEL.ONLINE_MODEL_PKG.GET_MODE_TWISS_DATA("+
                         "'" + q.instance() +"'" +
                        ",'" + posParam   +"'" +
                        "," + (typeParam.equals("NULL") ? "null" :
                              "'"+typeParam+"'") +
                        "," + (modeParam.equals("NULL") ? "null" :
                              "'"+modeParam+"'") +
                        "," + (runParam.equals("NULL") ? "null" :
                              "'"+runParam+"'") +
                        ")) T ORDER BY PSI_X";
{code}

runParam interpreted as: null = latest, 1 = is gold, > 1 = match to value

3) GET_MODE_RMATRIX_DATA

Expand permitted values of 5th param RunID, so that:
       null = latest (as now)

1 = gold
> 1 = match to argument (as now)

No change required to Aida's call to GET_MODE_RMATRIX_DATA, however,
runParam should now be interpreted as as above:

{code}
```
                    "SELECT R11, R12, R13, R14, R15, R16,"+
                    "R21, R22, R23, R24, R25, R26,"+
                    "R31, R32, R33, R34, R35, R36,"+
                    "R41, R42, R43, R44, R45, R46,"+
                    "R51, R52, R53, R54, R55, R56,"+
                    "R61, R62, R63, R64, R65, R66 FROM "+
                    "TABLE(MACHINE_MODEL.ONLINE_MODEL_PKG.GET_MODE_RMATRIX_DATA(" +
                     "'" + q.instance() +"'," +
                     "'" + posParam + "',"+
                     (typeParam.equals("NULL") ?
                        "null" : "'"+typeParam+"'") + "," +
                     (modeParam.equals("NULL") ?
                      "null" : "'"+modeParam+"'") + "," +
                     (runIdParam.equals("NULL") ?
                        "null" : "'"+runIdParam+"'") +
                     ")) T";
```
{code}

Model GUIs and APEX Reports

The following requirements apply to both the Model Optics GUI and to
the online Model upload and viewing web page [3].

Table of all Runs:

The table of model runs should be modified to include a new column
showing which model runs are gold. See example in [4] enclosed:

Designate which model is GOLD:

You must be able to designate which run of each mode is to be the gold
of that mode. This is separately assignable for each of extant and
design.

This should be done easily from the same screen as the Golds panel
 below.

Golds Screen: Display golds history and setting gold:

You will be able to see a history of which models have
previously been gold, as well as which is the present gold, of each
mode. See example in [5] enclosed.

[1] LCLS Model Data Provider Guide,
http://www.slac.stanford.edu/grp/cd/soft/aida/lclsModelDpGuide.html

[2] "MODE" handling requirements, http://tinyurl.com/da4fy9

[3] Model upload and viewing web page,
https://oraweb.slac.stanford.edu/apex/slacprod/f?p=400

[4] Example Run Data table, as in APEX and the Model GUI

gold_example_run_data.pdf

[5] Example Gold History, as would be included
in the model upload APEX application [3] and
Paul's new Model GUI:

gold_example_gold_report.pdf