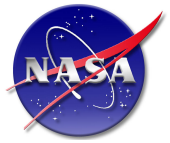


Trick 13 Simulation Environment: Tutorial Review

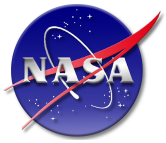
Hong Chen (L-3Com/ER7)

Alex Lin (NASA/ER7)

Donna Panter (L-3Com/ER7)

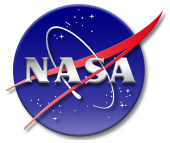
John Penn (L-3Com/ER7)

Warwick Woodard (L-3Com/ER7)

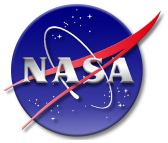


Tutorial Review Agenda

1. **Setting up the Environment**
2. **Introduction to the cannon ball (Trick-less)**
3. **Build a Trick cannon ball simulation**
4. **Run cannon ball simulation in real-time**
5. **Simulation Architecture (S_define syntax)**
6. **Input Processor (Python)**
7. **Viewing data with trick_dp**



Setting up the Environment



Set up the Environment

Objective

Setup Trick Environment

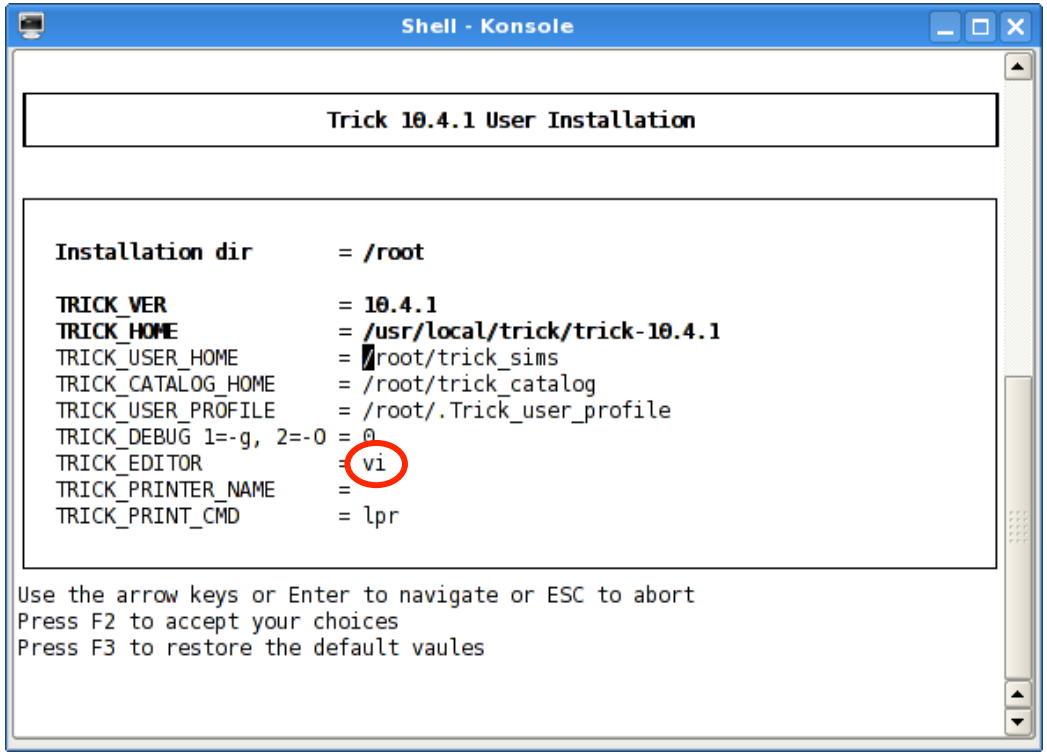
Prerequisites

Login credentials

Trick Training CD login automatic

Set up the Environment

```
% cd $HOME

% install_user
        Set TRICK_EDITOR to editor of choice (e.g. vim, emacs, nedit, etc.)

        Follow the on-screen instructions
```



Set up the Environment

- **csh/tcsh**

```
% vi .cshrc
    source $HOME/.Trick_cshrc_10.##.#
    (Where ##.# is the Trick version number)

% vi .Trick_user_cshrc
    setenv TRICK_CFLAGS "$TRICK_CFLAGS -Wall -g"      optional
    setenv TRICK_CFLAGS "$TRICK_CFLAGS -I$HOME/trick_models"

% source .cshrc
```

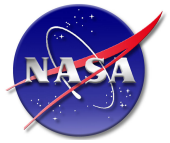
- bash

```
% vi .profile
    . $HOME/.Trick_profile_10.##.#

% vi .Trick_user_profile
    TRICK_CFLAGS="$TRICK_CFLAGS -Wall -g"             optional
    TRICK_CFLAGS="$TRICK_CFLAGS -I$HOME/trick_models"
    export TRICK_CFLAGS

% . .profile
```

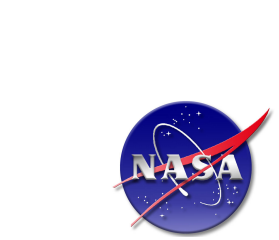
Trick Environment

- **Running "gte" should give you a list of Trick variables**
  - **Not all gte variables are in the environment**

```
% gte

TRICK_CATALOG_HOME=/root/trick_catalog
TRICK_CAT_MGR_HOME=/user/local/trick/trick-10.4.1/catalog
TRICK_CC=cc
TRICK_CFLAGS=-Wall -g -I/root/trick_models
TRICK_CPPC=c++
TRICK_CXXFLAGS=
TRICK_DEBUG=0
TRICK_EDITOR=vim
TRICK_EXEC_LINK_LIBS=
TRICK_FORCE_32BIT=0
TRICK_GTE_EXT=
TRICK_HOME=/user/local/trick/trick-10.4.1
TRICK_HOST_CPU=Linux_4.2_27
TRICK_HOST_CPU_USER_SUFFIX=
TRICK_HOST_TYPE=Linux
TRICK_ICG_EXCLUDE=
TRICK_MAKE=
TRICK_PATH=/user/local/trick/trick-10.4.1/bin_Linux_4.2_27:/user/local/trick/trick-10.4.1/bin
TRICK_PRINTER_NAME=
TRICK_PRINT_CMD=lpr
TRICK_USER_CSHRC=/root/.Trick_user_cshrc
TRICK_USER_HOME=/root/trick_sims
TRICK_USER_LINK_LIBS=
TRICK_USER_PROFILE=/root/.Trick_user_profile
TRICK_VER=10.0.dev
XML_CATALOG_FILES=/user/local/trick/trick-10.4.1/trick_source/data_products/DPX/XML/catalog.xml
```



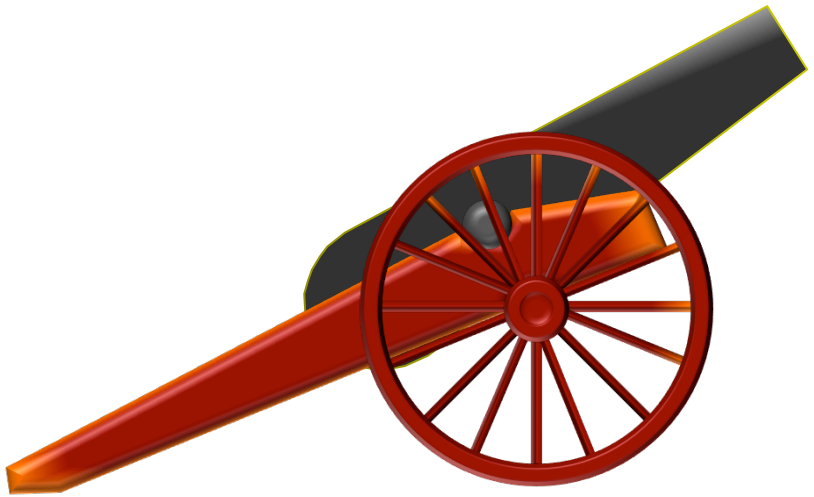
Introduction to the cannon ball

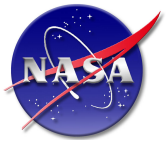


Trickless Cannonball

Objective :

Show a standalone C program to simulate a cannon ball



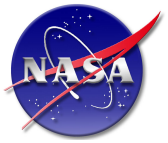


Cannonball Problem Statement

- **A cannonball at initial position (0,0) is shot at initial velocity (v_0) and initial angle (θ). How far does the cannonball travel?**

- **The analytical solution:**

	Horizontal	Vertical
Initial position	$p_{x0} = 0$	$p_{y0} = 0$
Initial velocity	$v_{x0} = v_0\cos(\theta)$	$v_{y0} = v_0\sin(\theta)$
Acceleration	$a_x = 0$	$a_y = -9.81$
Velocity	$v_x = v_{x0} + a_x t$	$v_y = v_{y0} + a_y t$
Position	$p_x = p_{x0} + v_{x0}t + \frac{1}{2}a_x t^2$	$p_y = p_{y0} + v_{y0}t + \frac{1}{2}a_y t^2$



Trickless Cannonball – Sample Problem

- **Example: cannon_ball.c**
  - **Standalone C program to simulate a cannon ball.**
  - **Program's components:**
    - Declarations
    - Default Data
    - Initialization
    - Executive
    - Shutdown

- **Declarations**

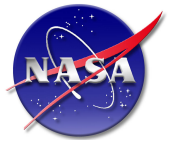
```
#include <stdio.h>
#include <math.h>

int main() {

        /* Declare variables used in simulation */
        double pos[2];          /* px and py */
        double pos_orig[2] ;    /* px0 and py0 */
        double vel[2];          /* vx and vy */
        double vel_orig[2] ;    /* vx0 and vy0 */
        double acc[2];          /* ax and ay */
        double init_angle ;     /* θ */
        double init_speed ;     /* v0 */
        double time ;
```

- **Default Data**

```
/* Initialize data */
acc[0] = 0.0 ;
acc[1] = -9.81 ;

time = 0.0 ;

init_angle = M_PI/6.0 ;
init_speed = 50.0 ;
```

- **Initialization**

```
/* Do initial calculations */
pos_orig[0] = 0 ;
pos_orig[1] = 0 ;
vel_orig[0] = cos(init_angle)*init_speed ;
vel_orig[1] = sin(init_angle)*init_speed ;
```



Trickless Cannonball - Executive

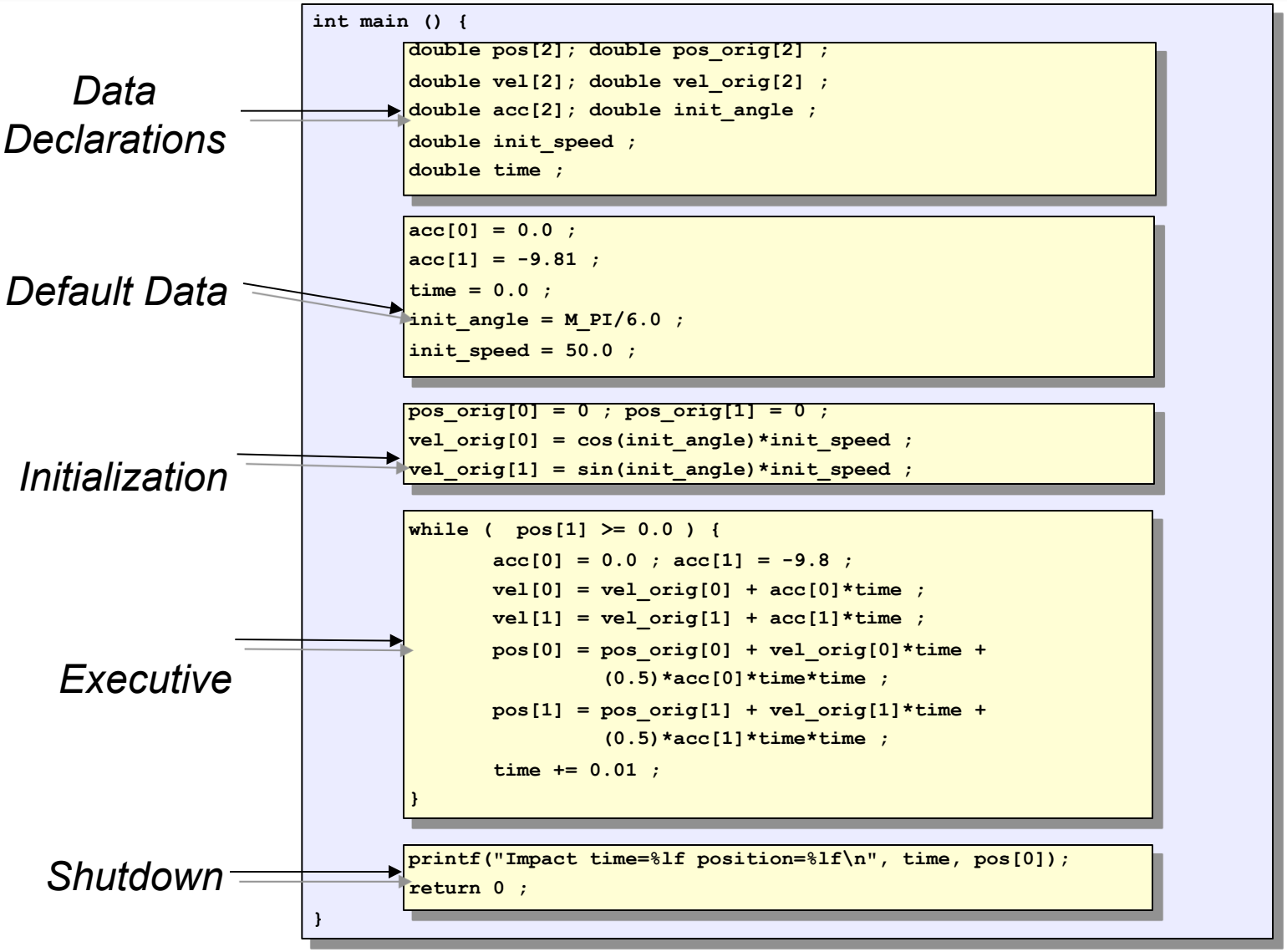
- **Executive**

```
/* Run simulation */
while (  pos[1] >= 0.0 ) {

        acc[0] = 0.0 ;
        acc[1] = -9.8 ;

        vel[0] = vel_orig[0] + acc[0]*time ;
        vel[1] = vel_orig[1] + acc[1]*time ;

        pos[0] = pos_orig[0] + vel_orig[0]*time +
                 (0.5)*acc[0]*time*time ;
        pos[1] = pos_orig[1] + vel_orig[1]*time +
                 (0.5)*acc[1]*time*time ;

        time += 0.01 ;
}
```

- **Shutdown**

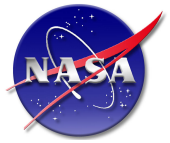
```
        /* Shutdown simulation */
        printf("Impact time=%lf position=%lf\n",
                time, pos[0]);

        return 0 ;
}
```

Full Standalone Simulation Components



- **Results**

```
% cc cannon_ball.c -lm

% ./a.out
Impact time=5.120000 position=221.269491
```



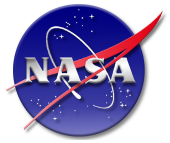
Trickless Cannonball - Shortcomings

- **There are some problems with the previous simulation**
  - **Not scalable or modular**
  - **No data recorded**
  - **No notion of real-time**
  - **Can't change initial state without recompiling**
  - **All variables are unitless**
  - **Position is a function of time (no state integration)**
  - **Cannonball impact inaccurate (impact event occurs between steps)**

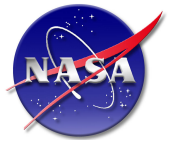
Trickless Cannon – Generalized To Trick

```
int main () {
    double pos[2]; double pos_orig[2] ;
    double vel[2]; double vel_orig[2] ;
    double acc[2]; double init_angle ;
    double init_speed ;
    double time ;

    pos[0] = 0.0 ; pos[1] = 0.0 ;
    vel[0] = 0.0 ; vel[1] = 0.0 ;
    acc[0] = 0.0 ; acc[1] = -9.81 ;
    time = 0.0 ;
    init_angle = M_PI/6.0 ; init_speed = 50.0 ;

    pos_orig[0] = pos[0] ; pos_orig[1] = pos[1] ;
    vel_orig[0] = cos(init_angle)*init_speed ;
    vel_orig[1] = sin(init_angle)*init_speed ;

    while (  pos[1] >= 0.0 ) {
           acc[0] = 0.0 ; acc[1] = -9.8 ;
           vel[0] = vel_orig[0] + acc[0]*time ;
           vel[1] = vel_orig[1] + acc[1]*time ;
           pos[0] = pos_orig[0] + vel_orig[0]*time +
                    (0.5)*acc[0]*time*time ;
           pos[1] = pos_orig[1] + vel_orig[1]*time +
                    (0.5)*acc[1]*time*time ;
           time += 0.01 ;
    }

    printf("Impact t=%lf pos=%lf\n", time, pos[0]);
    return 0 ;
}
```

- *Data declarations are in headers*
- *Defaults for data are specified in routines classed as "default_data"*
- *Initialization calculations occur in routines classed as "initialization"*
- *Executive functioning is managed by Trick and configurable through input files. Run-time routines are called by Trick's engine. And are ordered based on a user given class.*
- *Shutdown routines are called by Trick's executive after the main run-time loop.*



Build a Trick Cannon Ball Simulation

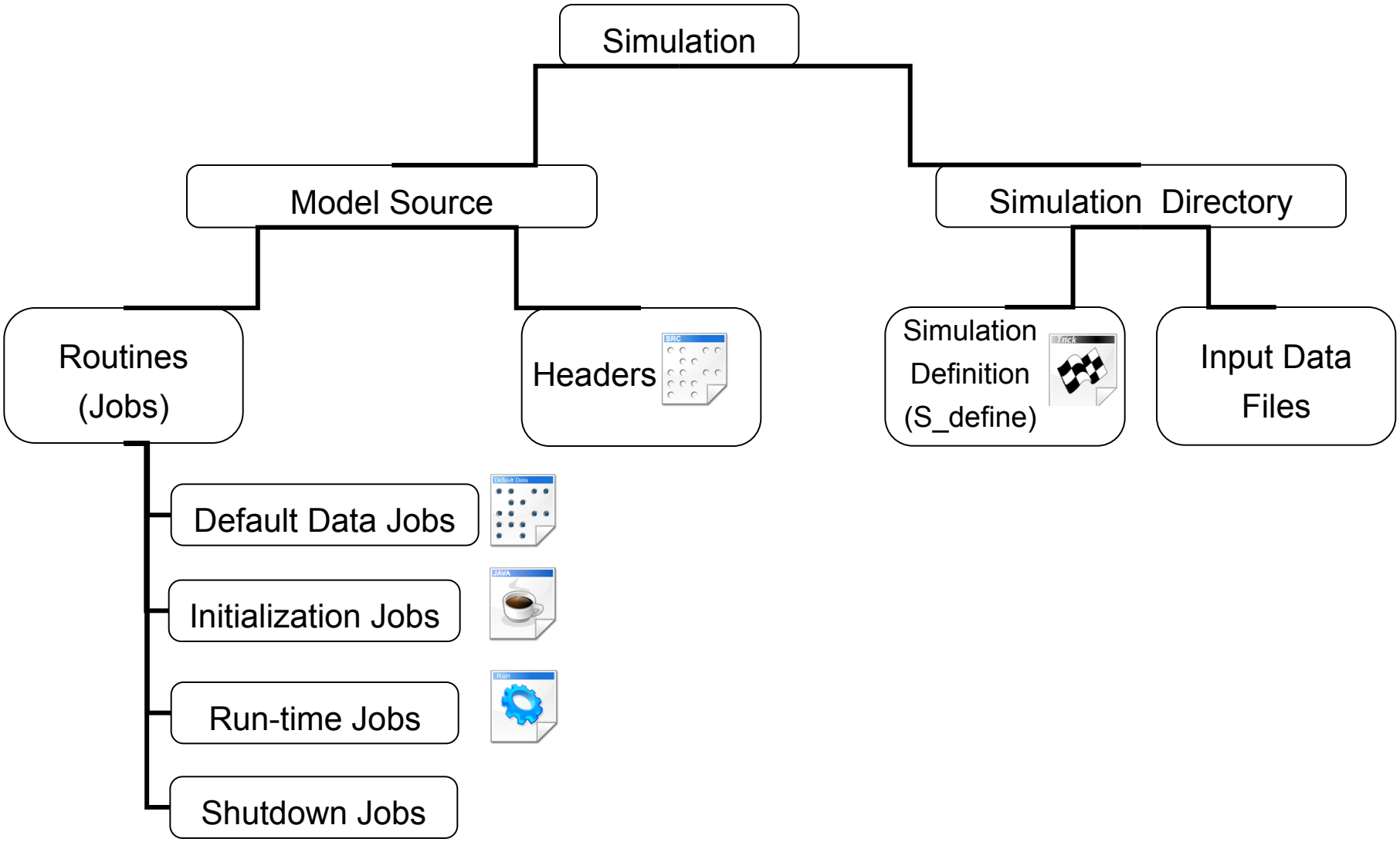


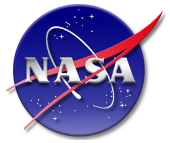
Build a Trick Simulation

- **Objective**
  - **Creating the directory system for cannon ball simulation**
  - **Build a "Trickified" cannon ball simulation**
  - **Putting the models together with the S_define**
  - **Creating a Run Input File**
  - **Adding Derivative and Integration Jobs**

- **Prerequisites**
  - **Trick environment set up correctly**
    - TRICK_CFLAGS must contain –I${HOME}/trick_models

Simulation Code Tree





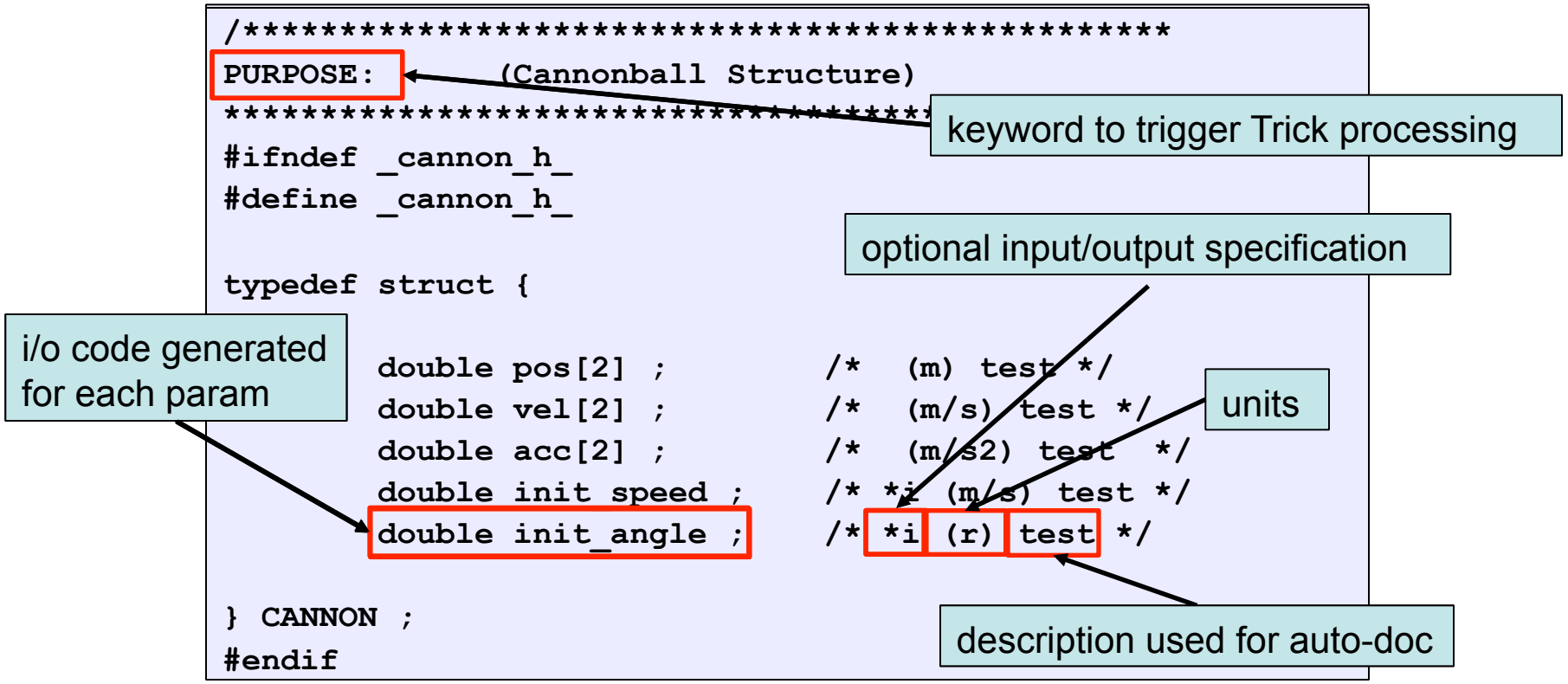
Creating A Directory System for Cannon Ball Simulation

```
% cd $HOME
% mkdir -p trick_sims/SIM_cannon_example
% mkdir -p trick_models/example/gravity/src
% mkdir -p trick_models/example/gravity/include
```

Note: It is standard practice, although not mandatory, to place sims in a "sims" directory, and model code in a "models" directory.

Data Declaration - cannon.h

- Declarations
  - Structures/Classes expected in header files (*.h)
  - Trick parses the header files keying on special comments
  - See section 3.4 of Trick tutorial for more information



Create cannon.h

```
% cd $HOME/trick_models/example/gravity/include
% vi cannon.h <edit as below & save>   OR
% cp $HOME/trick_models/copies/gravity/include/cannon.h .
```

```
/**************************************************************************
PURPOSE:                    (My first cannon test)
**************************************************************************/
#ifndef _cannon_h_
#define _cannon_h_
typedef struct {
 double pos0[2];   /* *i   (m)     Init position of cannonball */
 double vel0[2];   /* *i   (m/s)   Init velocity of cannonball */
 double acc0[2];   /* *i   (m/s2)  Init acceleration of cannonball */

 double pos[2];     /*     (m)     xy-position */
 double vel[2];     /*     (m/s)   xy-velocity */
 double acc[2];     /*     (m/s2)  xy-acceleration */
 double init_speed; /* *i (m/s)   Init barrel speed */
 double init_angle; /* *i (r)     Angle of cannon */

} CANNON;
#endif
```

Default Data Job – cannon_default_data()

- **Default Data**
  - *Default data no longer found in data files (*.d)*
  - *No automatic unit conversions with "{ }" notation*
  - *"default data" job classes called prior to initialization job classes*

```
/**************************************************
PURPOSE:      (Default data for cannonball)
**************************************************/
#include "../include/cannon.h"
#include "trick_utils/units/include/constant.h"


int cannon_default_data( CANNON* C ) {
    C->pos[0]      = 0.0 ;
    C->pos[1]      = 0.0 ;
    C->acc[0]      = 0.0 ;
    C->acc[1]      = -9.81 ;
    C->init_angle  = -30.0 * DTR ;
    C->init_speed  = 50.0 ;

    return 0 ;
}
```

Create cannon_default_data.c

```
% cd ../src
% vi cannon_default_data.c <edit as below and save> OR
% cp $HOME/trick_models/copies/gravity/src/cannon_default_data.c .
```

```
/**************************************************************************
PURPOSE: (Default data for cannonball)
**************************************************************************/
#include "../include/cannon.h"
#include "trick_utils/units/include/constant.h"

int cannon_default_data( CANNON* C)
{
  C->pos[0]      = 0.0 ;
  C->pos[1]      = 0.0 ;
  C->acc[0]      = 0.0 ;
  C->acc[1]      = -9.81 ;
  C->init_angle  = 30.0 * DTR ;
  C->init_speed  = 50.0 ;

  return 0 ;
}
```

Initialization Job – cannon_init()

- **Module files**
  - **Trick searches for header to use for auto documentation as well as "LIBRARY DEPENDENCIES"**
  - **Trick searches for function entry points**

```
/*************************************
 PURPOSE:      (Initialize the cannonball)
 LIBRARY_DEPENDENCIES: ((cannon_init.o))
*************************************************/
#include <math.h>
#include "../include/cannon.h"

int cannon_init(
      CANNON* C )
{
        C->vel[0] = C->init_speed*cos(C->init_angle);
        C->vel[1] = C->init_speed*sin(C->init_angle);

        return 0 ;
}
```

Keyword to trigger Trick processing
Trick header used for auto-doc

other objects not in the S_define that this module depends on (self dependency optional)

entry point

Create cannon_init.c

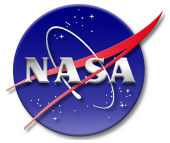
```
% vi cannon_init.c <edit as below and save> OR
% cp $HOME/trick_models/copies/gravity/src/cannon_init.c .
```

```
/**************************************************************
PURPOSE:                  (Initialize cannonball)
**************************************************************/
#include <stdio.h>
#include <math.h>
#include "../include/cannon.h"

int cannon_init( CANNON* C)
{
  C->pos0[0] = C->pos[0];
  C->pos0[1] = C->pos[1];

  C->vel[0] = C->init_speed * cos(C->init_angle);
  C->vel[1] = C->init_speed * sin(C->init_angle);
  C->vel0[0] = C->vel[0];
  C->vel0[1] = C->vel[1];

  C->acc0[0] = C->acc[0];
  C->acc0[1] = C->acc[1];
  return 0;
}
```

Create cannon_analytic.c

```
% vi cannon_analytic.c <edit as below and save> OR
% cp $HOME/trick_models/copies/gravity/src/cannon_analytic.c .
```

```
/************************TRICK HEADER*************************
PURPOSE: :                  (Initialize cannonball)
*************************************************************/
#include "../include/cannon.h"

int cannon_analytic( CANNON* C)
{
  static double time = 0.0;
  C->vel[0] = C->vel0[0] + C->acc0[0] * time;
  C->vel[1] = C->vel0[1] + C->acc0[1] * time;

  C->pos[0] = C->pos0[0] + C->vel0[0] * time + (0.5) * C->acc0[0] * time *
time;
  C->pos[1] = C->pos0[1] + C->vel0[1] * time + (0.5) * C->acc0[1] * time *
time;

  time += 0.01;

  return 0;
}
```

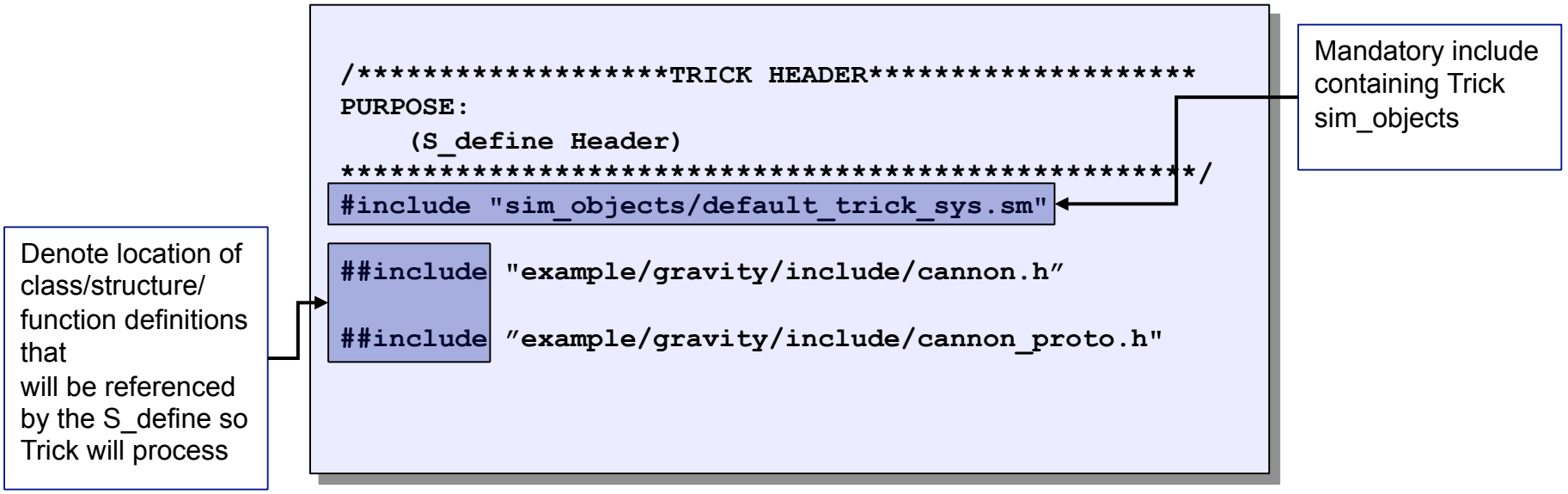


Simulation Definition - S_define

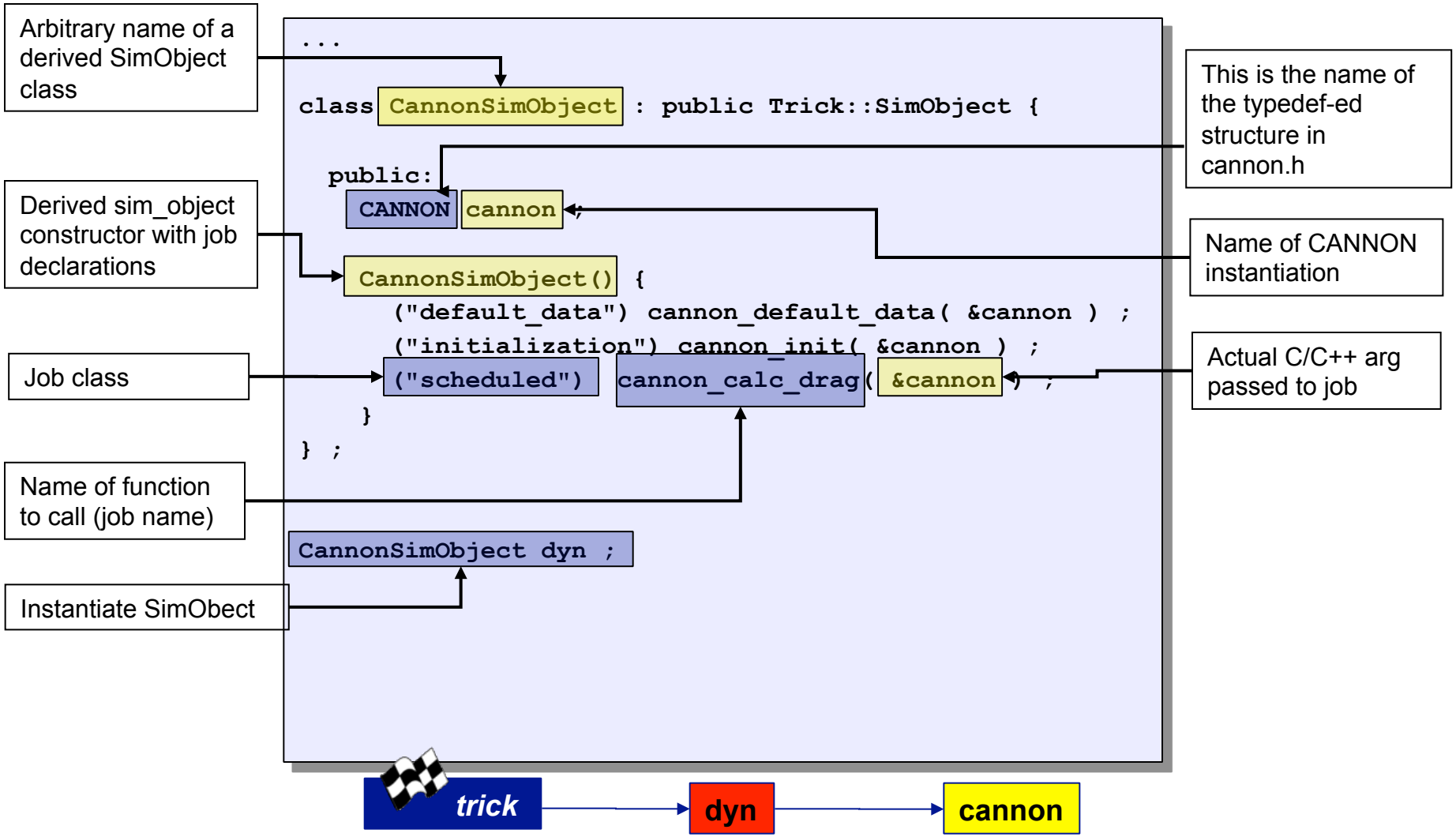
- **Trick uses a simulation definition file, or S_define, to pull all of the model pieces together into a simulation**

- **S_define contains**
  - **data structure instantiations**
  - **default data jobs to call**
  - **model jobs to call**
  - **model job frequencies**
  - **model job classes**
  - **importing/exporting data to other simulations**
  - **freeze cycles**
  - **integration frequencies**
  - **collect statements to gather a list of parameters into a single variable**

Simulation Definition - S_define



Simulation Definition – S_define (continued)



Create S_define for cannonball simulation

```
% cd $HOME/trick_sims/SIM_cannon_example
% vi S_define <edit as below and save>
```

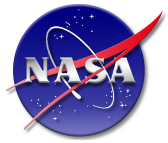
```
/**************************TRICK HEADER*************************
PURPOSE:(Simulate a cannon)
**************************************************************/

#include "sim_objects/default_trick_sys.sm"

##include "example/gravity/include/cannon.h"

##include "example/gravity/include/cannon_proto.h"
```

continued on next page

Create S_define for cannonball simulation (continued)

continued from previous page

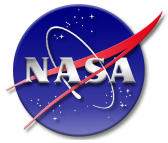
```
class CannonSimObject : public Trick::SimObject {

    public:
        CANNON cannon ;

        CannonSimObject() {
            ("default_data") cannon_default_data( &cannon ) ;
            ("initialization") cannon_init( &cannon ) ;
            (0.01, "scheduled") cannon_analytic( &cannon ) ;
        }
} ;
CannonSimObject dyn ;
```



cannon_proto.h

```
% cd $HOME/trick_models/example/gravity/include/
% vi cannon_proto.h <edit as below and save> OR
% cp $HOME/trick_models/copies/gravity/include/cannon_proto.h .
```

```
/******************************* TRICK HEADER ***************************
PURPOSE:          (CANNON PROTOTYPES HEADER)
LIBRARY_DEPENDENCIES: ((cannon_analytic.o) (cannon_init.o)
                       (cannon_default_data.o))
************************************************************************/
#include "cannon.h"

#ifdef __cplusplus /* If this is a C++ compiler, use C linkage */
extern "C" {
#endif

int cannon_analytic(CANNON*) ;
int cannon_init(CANNON*) ;
int cannon_default_data(CANNON*);

#ifdef __cplusplus
}
#endif
```



Compile the Cannonball Simulation

```
% cd $HOME/trick_sims/SIM_cannon_example
% CP
```

Abbreviated output to terminal

```
…
Generating S_sie.resource…
./S_main_Linux_4.2_27.exe sie
Created S_sie.resource file.
…

=== Simulation make complete ===
```

CP Auto Generated Files

CP auto-generates the following files:

S_source.cpp

This file contains all the model-specific simulation source code for run-time. Code that is common to all simulations can be found in: $TRICK_HOME

Makefile, Makefile_sim & Makefile_swig

This file contains all the Gnu-make rules for building and re-making the simulation

CP Auto Generated Files

CP auto-generated files (continued)

S_library_list

This file contains a list of the model files that CP processed

.auto_checksums

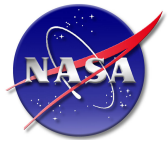
checksum calculation

S_sie.resource

This file contains XML formatted code describing all simulation variables (used for various Trick displays)

CP_out & MAKE_out

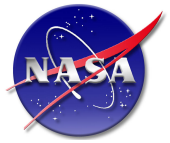
This file contains text output showing configuration processing (CP) step by step (also echoed to the screen when CP is executed)



Additional Auto Generated Files

- **CP calls several other autocode applications when building the simulation:**

- **ICG – Interface Code Generator**
  - **Generates header file I/O source code for use with Trick's memory management and data recording (ATTRIBUTES structure)**
  - **see io_src directories in model directories**



Input File

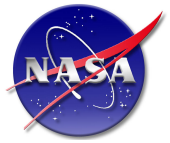
- Trick simulations can use a input file
- The input file is interpreted
  - No need to recompile the simulation after changing the file
- The syntax for the input file will be discussed later in the day.
- By convention, the input file is placed in a RUN_* directory

Create Input File & Run Simulation

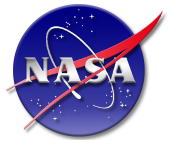
```
% mkdir RUN_test
% cd RUN_test
% vi input.py  <edit as below and save> OR
% cp $HOME/trick_sims/SIM_cannon_copy/RUN_test/input.py .
```

```
my_event = trick.new_event("impact")
my_event.set_cycle(0.01)
my_event.condition(0,"""trick.exec_get_sim_time() > 1.0 and \
                        dyn.cannon.pos[1] <= 0.0""")
my_event.action(0,"""print 'impact time: %f X-position: %s Y-position: %s' \
                    % (trick.exec_get_sim_time(), dyn.cannon.pos[0], \
                    dyn.cannon.pos[1])""")
trick.add_event(my_event)
my_event.activate()

trick.stop(5.2)
```

```
% cd ..
% ./S_main_*exe RUN_test/input.py
impact time: 5.110000 X-position: 220.836 m Y-position: -0.07905 m
```



Trick State Integration



Trick's Integration Capabilities

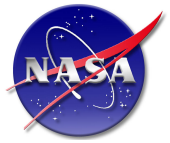
- Trick provides a **common interface** for different integration algorithms. Available algorithms are listed in section 7.11 of the User's Guide.

- Trick provides ***derivative*** and ***integration*** job classes to calculate the next simulation state from the previous.

- Trick calls the derivative and integration jobs multiple times, depending on the chosen integration algorithm. For example: Runge_Kutta_4 calls them four times for each simulation time step.

- Derivative jobs supply the derivatives to be integrated.

- Integration jobs :
  - Load the state and state derivatives into the integrator.
  - Call the integrate() function and then
  - Unload the results.
  - Tell the integration scheduler whether integration for the current time step is complete. That is: return the value returned by **integrate**().

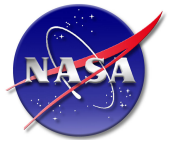


Cannonball using Trick's Integration Capabilities

In the cannonball simulation, the cannonball state : (velocity, position) will be periodically updated by integrating the following state derivatives (acceleration, velocity).

The acceleration will be fixed: (0, -9.81m/s2).

The velocity will obtained by the previous integration of acceleration.



cannon_deriv()

Derivative Job

```
/*********************************************
PURPOSE:          (Try Trick Integration)
*********************************************/
#include "../include/cannon.h"

int cannon_deriv(
   CANNON* C)
{
    C->acc[0] = 0.0 ;
    C->acc[1] = -9.81 ;

    return(0);
}
```

cannon_integ()

Integration Job

Load State

Integration Step

Unload State

Tell if we're done

```
/********************** TRICK HEADER**************************
PURPOSE:                (Cannon integration)
*************************************************************/
#include "sim_services/Integrator/include/integrator_c_intf.h"
#include "../include/cannon.h"
#include <stdlib.h>

int cannon_integ( CANNON *C)
{    int ipass ;

     load_state( &C->pos[0], &C->pos[1],
                 &C->vel[0], &C->vel[1], NULL);
     load_deriv( &C->vel[0], &C->vel[1],
                 &C->acc[0], &C->acc[1], NULL);

     ipass = integrate();

     unload_state( &C->pos[0], &C->pos[1],
                   &C->vel[0], &C->vel[1], NULL);

     return(ipass);
}
```

Create cannon_proto.h

```
% cd $HOME/trick_models/example/gravity/include
% vi cannon_proto.h <edit as below & save>
Change LIBRARY_DEPENDENCIES as shown, remove cannon_analytic(), and add
cannon_integ() and cannon_deriv().
```

```
/*************************************************************************
PURPOSE:      (Cannon Prototypes)
LIBRARY_DEPENDENCIES: ((cannon_integ.o) (cannon_deriv.o)
                       (cannon_init.o) (cannon_default_data.o))
*************************************************************************/
#ifndef _cannon_proto_h_
#define _cannon_proto_h_
#include <stdio.h>
#include "cannon.h"

#ifdef __cplusplus
extern "C" {
#endif

int cannon_integ(CANNON*) ;
int cannon_deriv(CANNON*) ;
int cannon_init(CANNON*) ;
int cannon_default_data(CANNON*) ;

#ifdef __cplusplus
}
#endif
#endif
```

Create cannon_deriv.c

```
% cd $HOME/trick_models/example/gravity/src
% vi cannon_deriv.c  <edit as below and save> OR
% cp $HOME/trick_models/copies/gravity/src/cannon_deriv.c .
```

```
/**************************************************************************
PURPOSE:                (Try Trick Integration)
**************************************************************************/
#include "../include/cannon.h"

int cannon_deriv ( CANNON* C)
{
  C->acc[0] = 0.0;
  C->acc[1] = -9.81;

  return 0;
}
```

Create cannon_integ.c

```
% vi cannon_integ.c  <edit as below and save> OR
% cp $HOME/trick_models/copies/gravity/src/cannon_integ.c .
```

```
/********************************** TRICK HEADER *************************
PURPOSE:                        (Cannon integration)
*************************************************************************/
#include "sim_services/Integrator/include/integrator_c_intf.h"
#include "../include/cannon.h"
#include <stdlib.h>

int cannon_integ( CANNON *C)
{
     int ipass ;
     load_state( &C->pos[0], &C->pos[1],
                 &C->vel[0], &C->vel[1],
                 NULL);

     load_deriv( &C->vel[0], &C->vel[1] ,
                 &C->acc[0], &C->acc[1] ,
                 NULL);

     ipass = integrate();

     unload_state( &C->pos[0], &C->pos[1] ,
                   &C->vel[0], &C->vel[1] ,
                   NULL);

     return(ipass);
}
```

Update S_define

```
% cd $HOME/trick_sims/SIM_cannon_example
% vi S_define <edit as below and save>
```

```
/************************** Trick Header *********************************
PURPOSE: (S_define Header)
*************************************************************************/
#include "sim_objects/default_trick_sys.sm"
##include "example/gravity/include/cannon.h"
##include "example/gravity/include/cannon_proto.h"

class CannonSimObject : public Trick::SimObject {
    public:
    CANNON cannon ;

    CannonSimObject() {
        ("default_data") cannon_default_data( &cannon ) ;
        ("initialization") cannon_init( &cannon ) ;
        (0.01, "scheduled") cannon_analytic( &cannon ) ;
        ("derivative") cannon_deriv(&cannon) ;
        ("integration") trick_ret = cannon_integ(&cannon) ;
    }
} ;
CannonSimObject dyn ;
IntegLoop dyn_integloop (0.01) dyn;
```

Explanation of S_define Updates

```
class CannonSimObject : public Trick::SimObject {
    public:
    CANNON cannon ;

    CannonSimObject() {
        ("default_data") cannon_default_data( &cannon ) ;
        ("initialization") cannon_init( &cannon ) ;

        ("derivative") cannon_deriv(&cannon) ;
        ("integration") trick_ret = cannon_integ(&cannon) ;
    }
} ;
CannonSimObject dyn ;
IntegLoop dyn_integloop (0.01) dyn;
```

Integration loop

Unique name

Integration rate

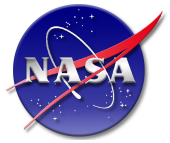
List of sim objects whose integration this IntegLoop Controls. (comma delimited)

Add the jobs required for integration.

Instantiate an Integrator Object

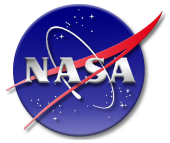
```
% cd $HOME/trick_sims/SIM_cannon_example/RUN_test
% vi input.py <edit as below and save>
```

```
dyn_integloop.getIntegrator(trick.Runge_Kutta_4, 4)
my_event = trick.new_event("impact")
my_event.set_cycle(0.01)
my_event.condition(0,"""trick.exec_get_sim_time() > 1.0 and \
                        dyn.cannon.pos[1] <= 0.0""")
my_event.action(0,"""print 'impact time: %f X-position: %s Y-position: %s' \
                   % (trick.exec_get_sim_time(), dyn.cannon.pos[0], \
                   dyn.cannon.pos[1])""")
trick.add_event(my_event)
my_event.activate()

trick.stop(5.2)
```

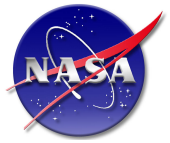


Re-Compile Trick Simulation and Run

```
% cd ..
% make clean

% CP
% ./S_main_*exe RUN_test/input.py
impact time: 5.100000 X-position: 220.836 m Y-position: -0.07905 m
```

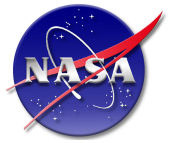


Running Cannonball Simulation in Real Time



Real Time

- Objective
  - Create a real-time input file
  - Get-to-know the Simulation Control Panel
- Prerequisites
  - Trick environment set up correctly
    - TRICK_CFLAGS must contain –I${HOME}/trick_models
  - Previous section's cannon ball simulation compiles and runs



Create Real Time Input File

```
% cd $HOME/trick_sims/SIM_cannon_example
% mkdir Modified_data
% cd Modified_data
% vi realtime.py <edit as below and save> OR
% cp $HOME/trick_sims/SIM_cannon_copy/Modified_data/realtime.py .

trick.frame_log_on()
trick.real_time_enable()
trick.exec_set_software_frame(0.1)
trick.itimer_enable()

trick.exec_set_enable_freeze(True)
trick.exec_set_freeze_command(True)
trick.sim_control_panel_set_enabled(True)
```

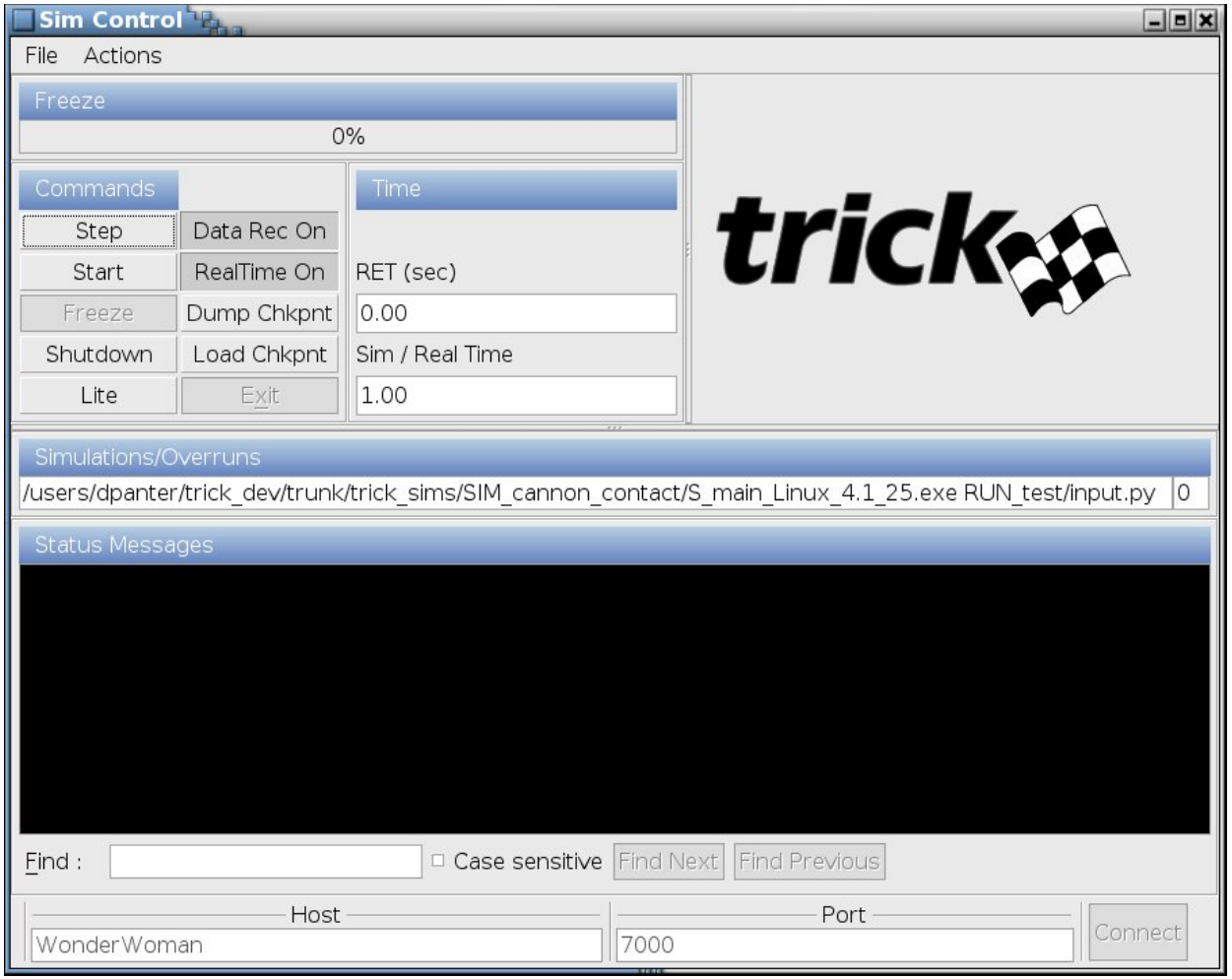
Update Input File and Run Executable

```
% cd ../RUN_test
% vi input.py <edit as below and save>


execfile("Modified_data/realtime.py")

.
.
.

trick.stop(5.2)
```

```
% cd ..
% ./S_main*exe RUN_test/input.py &
```

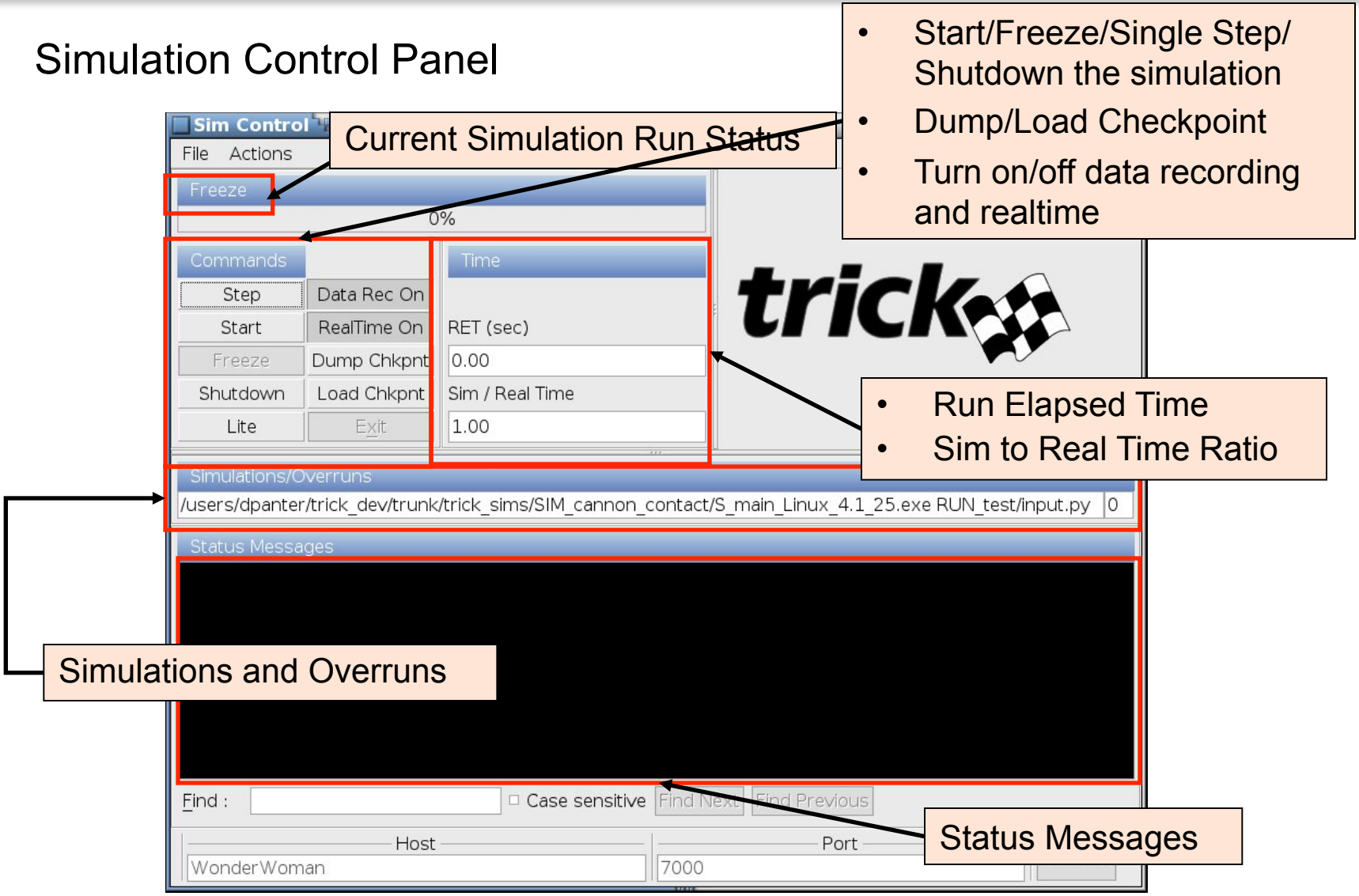
Running Simulation in Real-Time

- Simulation Control Panel



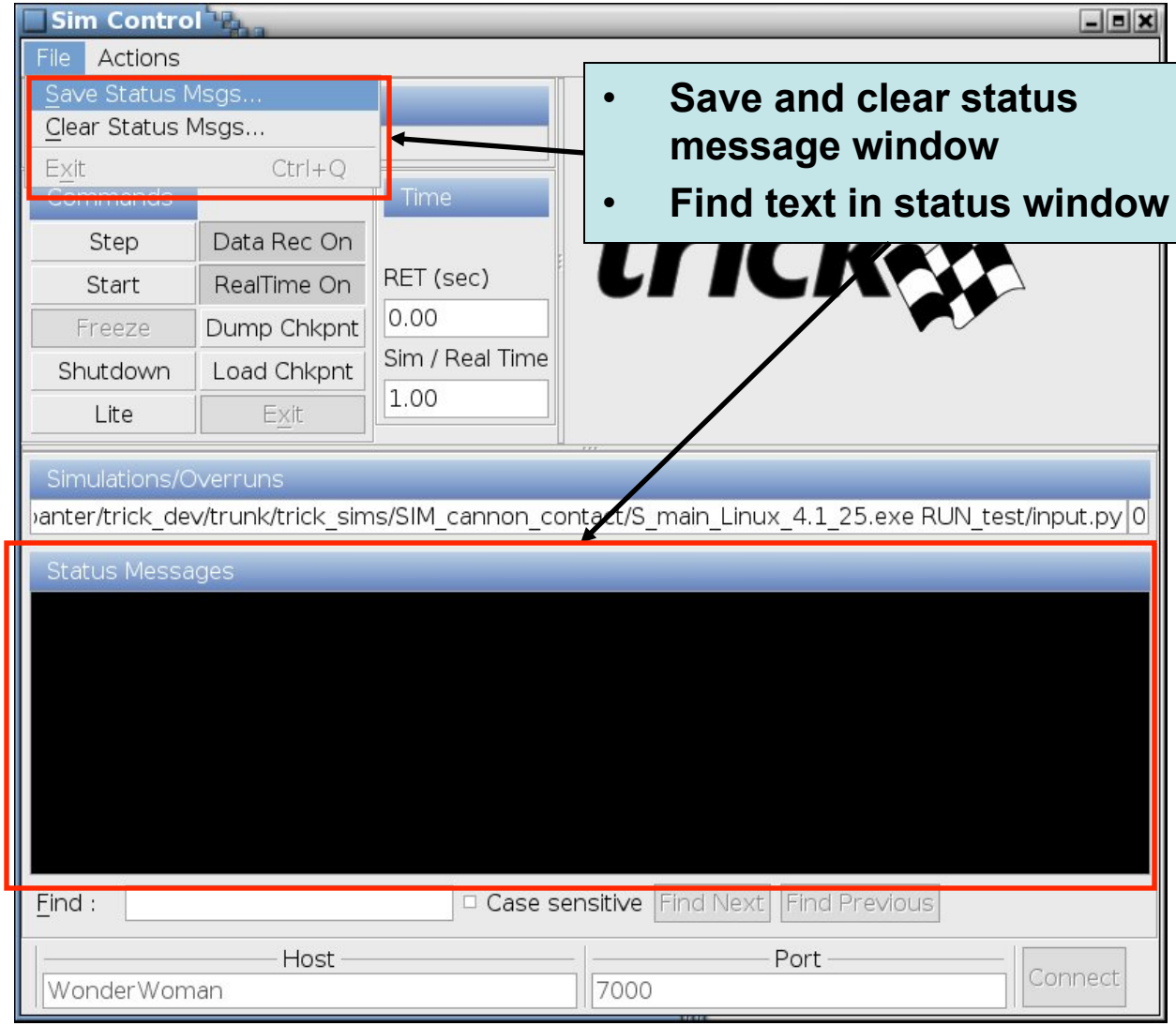
Running Simulation in Real-Time

- Simulation Control Panel



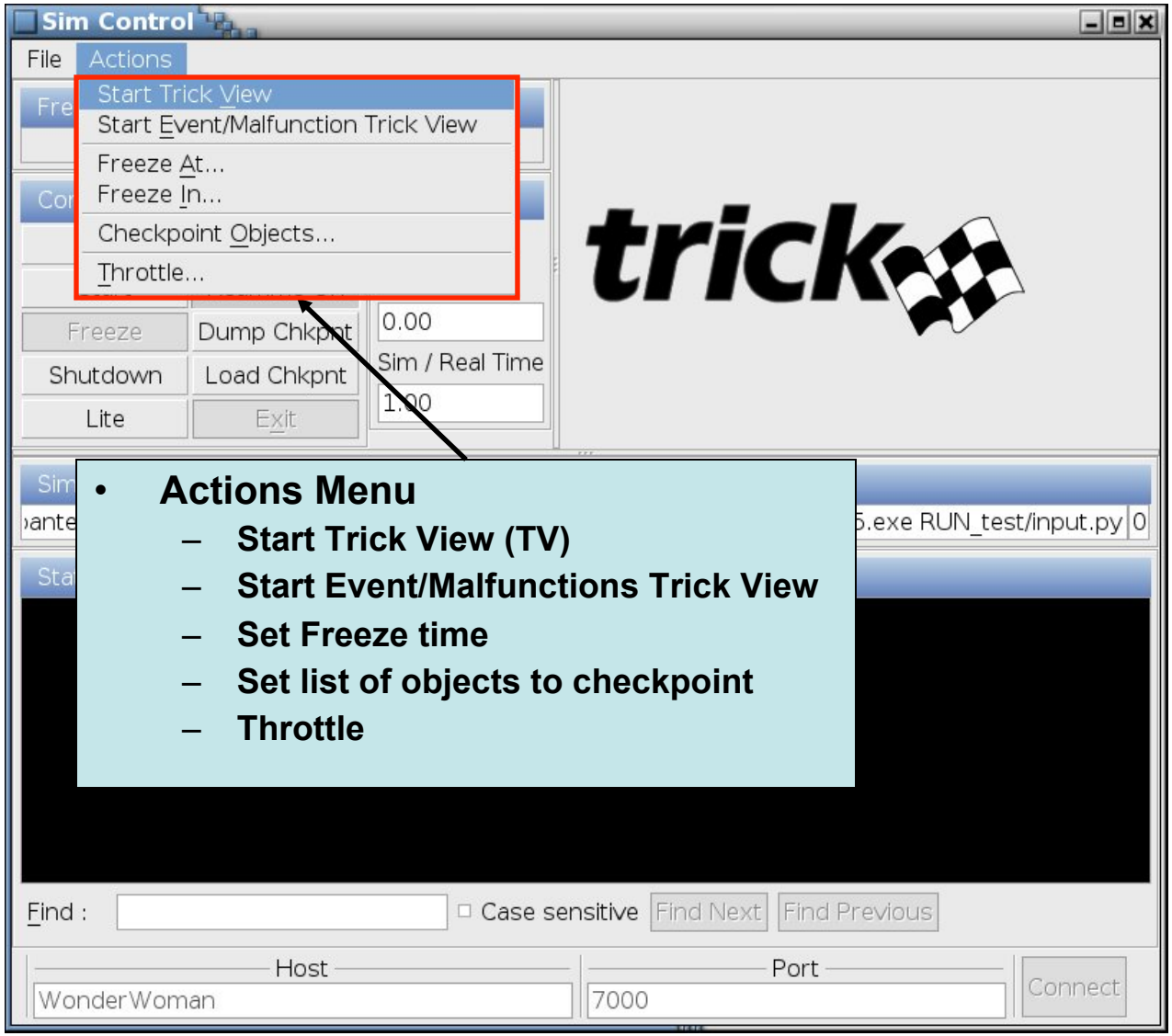
Running Simulation in Real-Time

- File menu

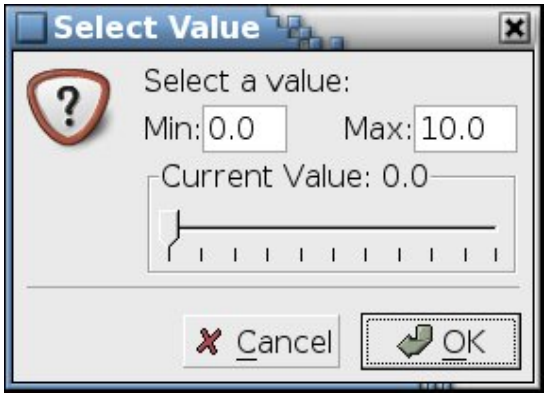


Running Simulation in Real-Time

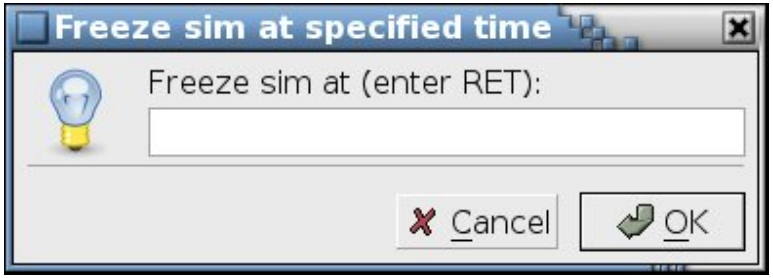
- Actions Menu



- Throttle
  - Only available when itimers are off
    - See real time presentations and User's Guide for itimer information
  - Allows sim to run a multiple of realtime

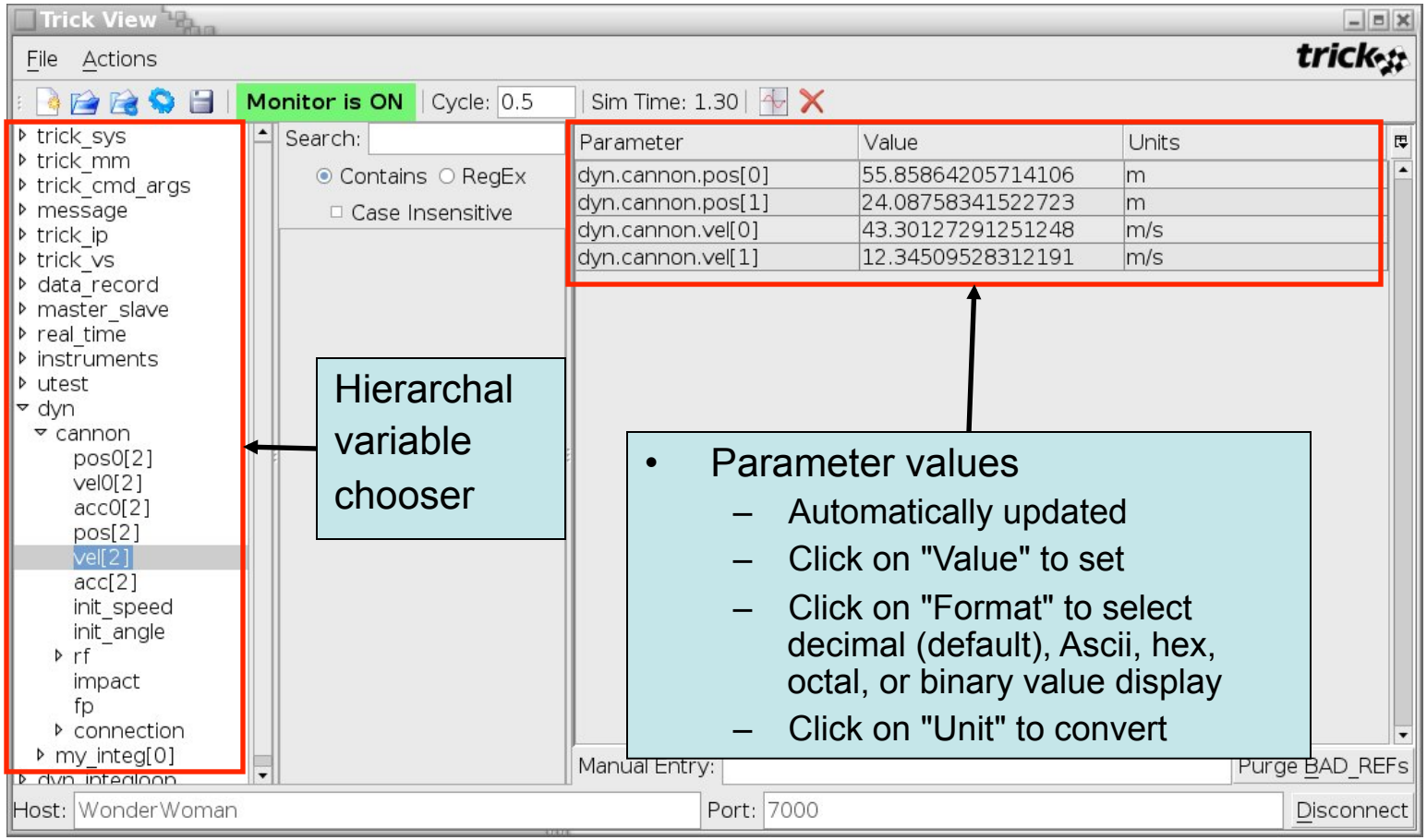


- Freeze Popup
  - Freeze at = absolute time
  - Freeze in = relative time



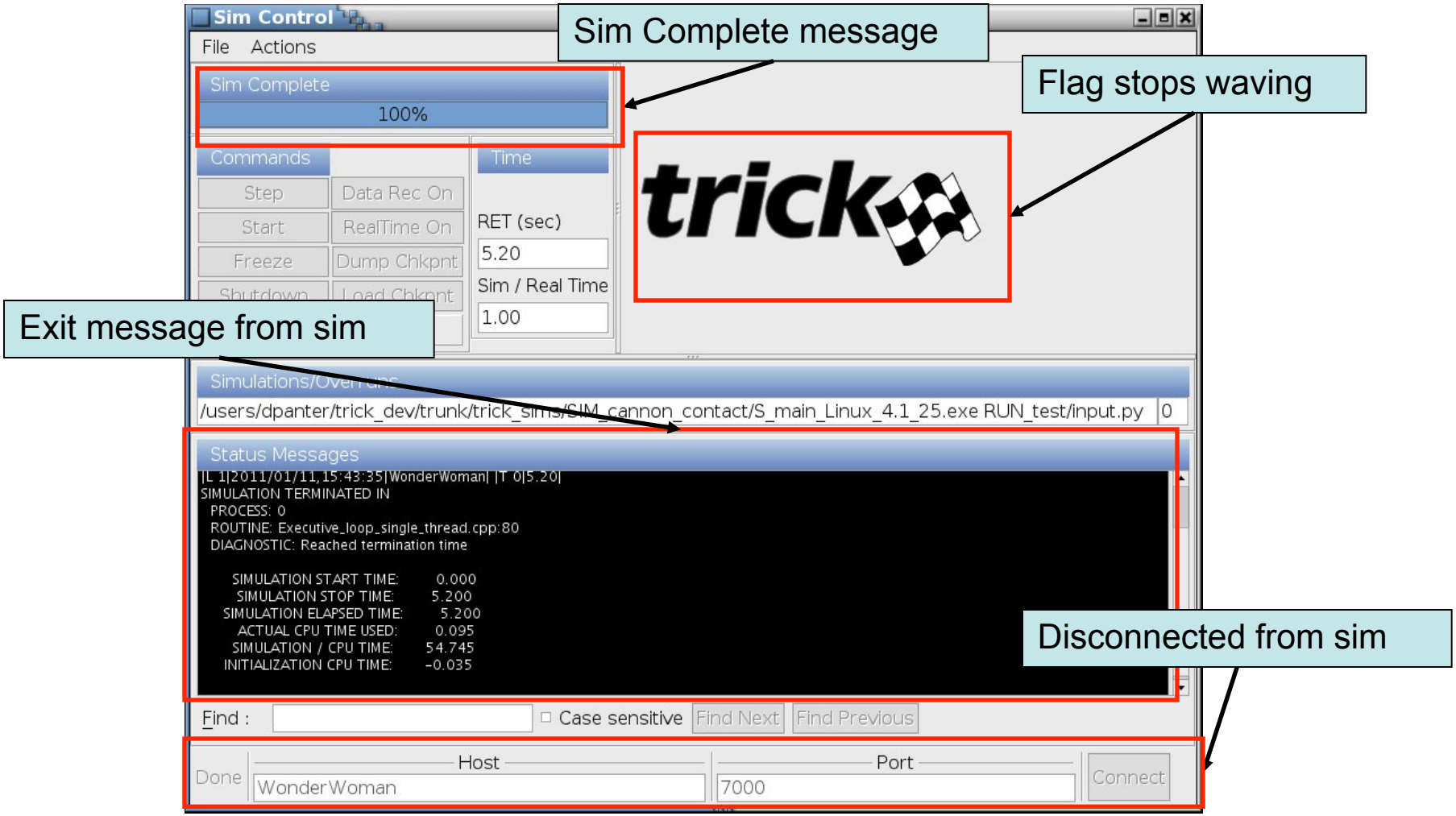
Running Simulation in Real-Time

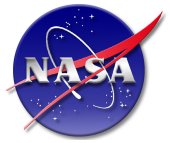
- Trick View (TV)
  - View/Set variables within the simulation
    - Double-click variable in variable tree to add to parameter table



Running Simulation in Real-Time

- Panel indicating simulation complete





Simulation Architecture

- **Objective**
  - **Job Scheduling**
    - Job Classes
    - Job Frequencies and offsets
    - Phasing
    - Threading simulations
    - Integration and Collect statements

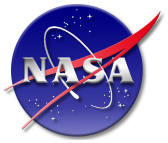
- **Prerequisites**
  - **Trick environment set up correctly**
    - TRICK_CFLAGS must contain –I${HOME}/trick_models

Job Classes

- **Jobs classes**
  - **Each S_define level job is required to have a job class**
  - **The job class determines the order of the module calls when the job is scheduled to run**
  - **Trick has many different job classes**

```
class CannonSimObject : public Trick::SimObject {
    public:
        CANNON cannon ;

        CannonSimObject() {
            ("default_data") cannon_default_data( &cannon ) ;

            ("initialization") cannon_init( &cannon ) ;

            ("derivative") cannon_deriv(&cannon) ;

            ("integration") trick_ret = cannon_integ(&cannon) ;
        }
} ;
CannonSimObject dyn ;
```

- **Job Classes - Initialization**
  - **`"default_data"`**
    - Module executed only once.
    - The only class called before the input file is read.
  - **`"initialization"`**
    - Module executed only once, at simulation time = 0. Is called after the input file is read.
  - **`"restart"`**
    - Run once after a checkpoint file is loaded or sim is started from checkpoint file.



Job Classes

- **Job Classes – Executive**
  - **`"environment"`** (e.g., atmosphere, third body gravitation)
  - **`"sensor"`** (e.g., Gyros, Accelerometers, Vision)
  - **`"sensor_emitter"`** (e.g., radar, laser)
  - **`"sensor_reflector"`** (e.g., surfaces)
  - **`"sensor_receiver"`** (e.g., radar receiver, laser receiver)
  - **`"scheduled"`** (Default scheduled job class)
  - **`"effector"`** (e.g., RCS, servo motors, flaps)
  - **`"effector_emitter"`** (e.g., plume, active magnetic fields)
  - **`"effector_receiver"`** (e.g., experiences effects from receivers)
  - **`"automatic_last"`** (e.g., self scheduling, but runs last)
  - **`"logging"`** (Sim data logging functions, Trick internal job class)
  - **`"end_of_frame"`**

most common executive class

- **Trick executes all of these jobs in the same loop in the above order. Their position in the list is what really distinguishes these job classes.**

- **Job Classes – Integration**
  - **`"derivative"`**
    - Equations of motion (EOM) derivative function.
  - **`"integration"`**
    - EOM state integration function.
  - **`"dynamic_event"`**
    - Provides a continuous time dependent equation whose root defines a discontinuous event in the system EOM. Evaluation of function returns an estimated delta time to reach the root.
  - **`"post_Integration"`**
    - Runs after the integration loop is finished
- **These job classes are not run at simulation time = 0. The exception is that a derivative job can be configured to run at time 0.**

- **Job Classes – Freeze and Checkpointing**
  - **`"freeze_init"`**
    - Run once when entering freeze.
  - **`"freeze"`**
    - Cyclically called while in freeze.
  - **`"unfreeze"`**
    - Run once when returning to Run.
  - **`"checkpoint"`**
    - Run before a checkpoint is taken.
  - **`"preload_checkpoint"`**
    - Run before a checkpoint is loaded.
  - **`"post_checkpoint"`**
    - Run after a checkpoint is loaded.

- **Job Classes – Other**
  - **`"automatic"`**
    - Self scheduling job class. Job is expected to reschedule itself via its job control inputs.
  - **`"random"`**
    - Execution occurs at a specified delta time plus or minus 1 sigma random time.
  - **`"shutdown"`**
    - Run when sim is exiting.

- **Job Classes – Monte Carlo**
  - **`"monte_master_init"`**
    - Runs when master sim is initialized
  - **`"monte_master_pre"`**
    - Runs before new data is dispatched to slave sim
  - **`"monte_master_post"`**
    - Runs after result is returned from slave
  - **`"monte_master_shutdown"`**
    - Runs when master shuts down
  - **`"monte_slave_init"`**
    - Runs when slave sim is initialized
  - **`"monte_slave_pre"`**
    - Runs after new data is received from master
  - **`"monte_slave_post"`**
    - Runs after slave sim is completed (sends result to master)
  - **`"monte_slave_shutdown"`**
    - Runs when monte carlo master comm is lost and slave shuts down

Job Frequencies

- **Derivative and Integration jobs inherit their frequencies from the IntegLoop statement**

```
class CannonSimObject : public Trick::SimObject {
    public:
        CANNON cannon ;

        CannonSimObject() {
            ("initialization") cannon_init( &cannon ) ;
            ("default_data") cannon_default_data( &cannon ) ;
            ("derivative")                cannon_deriv(&cannon) ;
            ("integration") trick_ret = cannon_integ(&cannon) ;
            (0.015, "scheduled") cannon_calc_drag( &cannon ) ;
        }
}
CannonSimObject dyn ;
IntegLoop dyn_integloop (0.01) dyn;
```

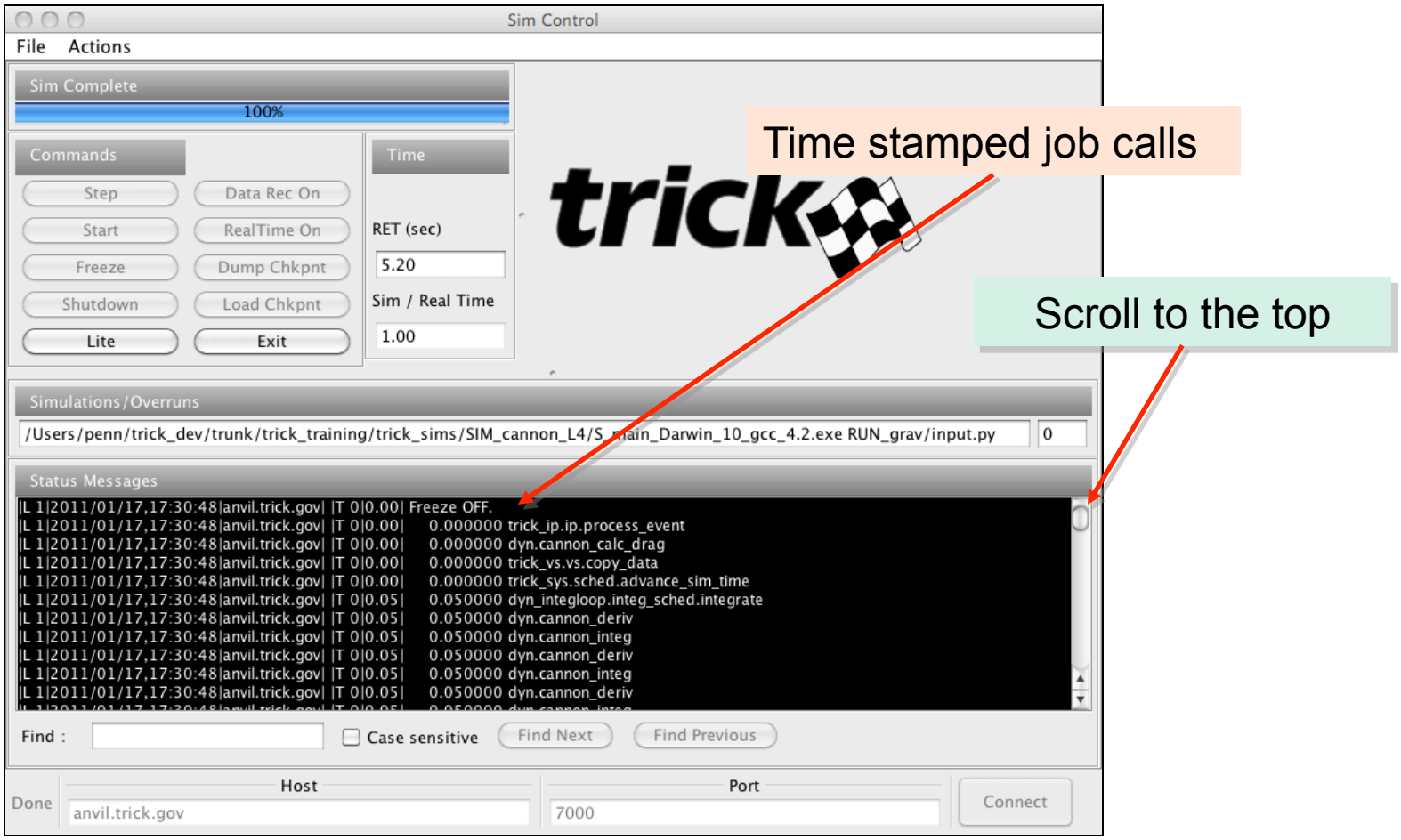
SIM_cannon_L4

To illustrate job frequencies let's run SIM_cannon_L4

```
% cd $HOME/trick_sims/SIM_cannon_L4
% CP
% ./S_main_${TRICK_HOST_CPU}.exe RUN_grav/input.py
```

SIM_cannon_L4

Job echoing was turned on with trick.echo_jobs_on().



Analyze Echo Job Output

```
Various init jobs above
0.000000 dyn.cannon_deriv
0.000000 trick_ip.ip.process_event
0.000000 dyn.cannon_calc_drag
0.000000 trick_vs.vs.copy_data
0.000000 trick_sys.sched.advance_sim_time

0.010000 dyn_integloop.integ_sched.integrate
0.010000 dyn.cannon_deriv
0.010000 dyn.cannon_integ
0.010000 dyn.cannon_deriv
0.010000 dyn.cannon_integ

0.015000 dyn.cannon_calc_drag

0.020000 dyn_integloop.integ_sched.integrate
0.020000 dyn.cannon_deriv
0.020000 dyn.cannon_integ
0.020000 dyn.cannon_deriv
0.020000 dyn.cannon_integ

0.030000 dyn_integloop.integ_sched.integrate
0.030000 dyn.cannon_deriv
0.030000 dyn.cannon_integ
0.030000 dyn.cannon_deriv
0.030000 dyn.cannon_integ
0.030000 dyn.cannon_calc_drag
```

- **The time stamp shown is the simulation time, not "wall clock" time**

Job Frames

```
Various init jobs above
0.000000 dyn.cannon_deriv
0.000000 trick_ip.ip.process_event
0.000000 dyn.cannon_calc_drag
0.000000 trick_vs.vs.copy_data
0.000000 trick_sys.sched.advance_sim_time

0.010000 dyn_integloop.integ_sched.integrate
0.010000 dyn.cannon_deriv
0.010000 dyn.cannon_integ
0.010000 dyn.cannon_deriv
0.010000 dyn.cannon_integ

0.015000 dyn.cannon_calc_drag

0.020000 dyn_integloop.integ_sched.integrate
0.020000 dyn.cannon_deriv
0.020000 dyn.cannon_integ
0.020000 dyn.cannon_deriv
0.020000 dyn.cannon_integ

0.030000 dyn_integloop.integ_sched.integrate
0.030000 dyn.cannon_deriv
0.030000 dyn.cannon_integ
0.030000 dyn.cannon_deriv
0.030000 dyn.cannon_integ
0.030000 dyn.cannon_calc_drag
```

- Trick cyclically creates a queue of jobs and executes them in an order determined by job class and S_define order
- Trick prepares a job queue at times when any job needs to execute
- Higher frequency jobs mostly determine the granularity of job queue creation
- Job frames have no notion of real-time synchronization

Job Frames (Initialization)

```
Various init jobs above
0.000000 dyn.cannon_deriv
0.000000 trick_ip.ip.process_event
0.000000 dyn.cannon_calc_drag
0.000000 trick_vs.vs.copy_data
0.000000 trick_sys.sched.advance_sim_time

0.010000 dyn_integloop.integ_sched.integrate
0.010000 dyn.cannon_deriv
0.010000 dyn.cannon_integ
0.010000 dyn.cannon_deriv
0.010000 dyn.cannon_integ

0.015000 dyn.cannon_calc_drag

0.020000 dyn_integloop.integ_sched.integrate
0.020000 dyn.cannon_deriv
0.020000 dyn.cannon_integ
0.020000 dyn.cannon_deriv
0.020000 dyn.cannon_integ

0.030000 dyn_integloop.integ_sched.integrate
0.030000 dyn.cannon_deriv
0.030000 dyn.cannon_integ
0.030000 dyn.cannon_deriv
0.030000 dyn.cannon_integ
0.030000 dyn.cannon_calc_drag
```

- **Initialization**
- **During initialization, initialization and derivative class jobs are called**
- **Initialization jobs may be ordered with the S_define P# syntax**

Job Frames (Time Zero)

```
Various init jobs above
0.000000 dyn.cannon_deriv
0.000000 trick_ip.ip.process_event
0.000000 dyn.cannon_calc_drag
0.000000 trick_vs.vs.copy_data
0.000000 trick_sys.sched.advance_sim_time

0.010000 dyn_integloop.integ_sched.integrate
0.010000 dyn.cannon_deriv
0.010000 dyn.cannon_integ
0.010000 dyn.cannon_deriv
0.010000 dyn.cannon_integ

0.015000 dyn.cannon_calc_drag

0.020000 dyn_integloop.integ_sched.integrate
0.020000 dyn.cannon_deriv
0.020000 dyn.cannon_integ
0.020000 dyn.cannon_deriv
0.020000 dyn.cannon_integ

0.030000 dyn_integloop.integ_sched.integrate
0.030000 dyn.cannon_deriv
0.030000 dyn.cannon_integ
0.030000 dyn.cannon_deriv
0.030000 dyn.cannon_integ
0.030000 dyn.cannon_calc_drag
```

Job Frame 1

- Integration class jobs are not called at time=0.0
- ALL other run-time jobs are called at time zero. If this is undesired, use an offset

Job Frames (Second)

```
Various init jobs above
0.000000 dyn.cannon_deriv
0.000000 trick_ip.ip.process_event
0.000000 dyn.cannon_calc_drag
0.000000 trick_vs.vs.copy_data
0.000000 trick_sys.sched.advance_sim_time

0.010000 dyn_integloop.integ_sched.integrate
0.010000 dyn.cannon_deriv
0.010000 dyn.cannon_integ
0.010000 dyn.cannon_deriv
0.010000 dyn.cannon_integ

0.015000 dyn.cannon_calc_drag

0.020000 dyn_integloop.integ_sched.integrate
0.020000 dyn.cannon_deriv
0.020000 dyn.cannon_integ
0.020000 dyn.cannon_deriv
0.020000 dyn.cannon_integ

0.030000 dyn_integloop.integ_sched.integrate
0.030000 dyn.cannon_deriv
0.030000 dyn.cannon_integ
0.030000 dyn.cannon_deriv
0.030000 dyn.cannon_integ
0.030000 dyn.cannon_calc_drag
```

Job Frame 2

- Integration class jobs follow derivative class jobs at the top of the job frame. If they are in a job queue, they are executed first.
- In this example, derivative and integration jobs are called twice. This is because we are running with Runge Kutta 2 which requires two passes.
- cannon_calc_drag() is not queued since its freq doesn't match the 0.010

Job Frames (Third)

Various init jobs above
0.000000 dyn.cannon_deriv
0.000000 trick_ip.ip.process_event
0.000000 dyn.cannon_calc_drag
0.000000 trick_vs.vs.copy_data
0.000000 trick_sys.sched.advance_sim_time

0.010000 dyn_integloop.integ_sched.integrate
0.010000 dyn.cannon_deriv
0.010000 dyn.cannon_integ
0.010000 dyn.cannon_deriv
0.010000 dyn.cannon_integ

0.015000 dyn.cannon_calc_drag

0.020000 dyn_integloop.integ_sched.integrate
0.020000 dyn.cannon_deriv
0.020000 dyn.cannon_integ
0.020000 dyn.cannon_deriv
0.020000 dyn.cannon_integ

0.030000 dyn_integloop.integ_sched.integrate
0.030000 dyn.cannon_deriv
0.030000 dyn.cannon_integ
0.030000 dyn.cannon_deriv
0.030000 dyn.cannon_integ
0.030000 dyn.cannon_calc_drag

Job Frame 3

- No other jobs have frequencies that fall on 0.015, so cannon_calc_drag() is alone in the job queue
- The speed of job queue creation is dependent on the number of jobs available. Bear this in mind before making a job with a weird frequency that forces continual job queueing

Job Frames (Fourth)

Various init jobs above
0.000000 dyn.cannon_deriv
0.000000 trick_ip.ip.process_event
0.000000 dyn.cannon_calc_drag
0.000000 trick_vs.vs.copy_data
0.000000 trick_sys.sched.advance_sim_time

0.010000 dyn_integloop.integ_sched.integrate
0.010000 dyn.cannon_deriv
0.010000 dyn.cannon_integ
0.010000 dyn.cannon_deriv
0.010000 dyn.cannon_integ

0.015000 dyn.cannon_calc_drag

0.020000 dyn_integloop.integ_sched.integrate
0.020000 dyn.cannon_deriv
0.020000 dyn.cannon_integ
0.020000 dyn.cannon_deriv
0.020000 dyn.cannon_integ

0.030000 dyn_integloop.integ_sched.integrate
0.030000 dyn.cannon_deriv
0.030000 dyn.cannon_integ
0.030000 dyn.cannon_deriv
0.030000 dyn.cannon_integ
0.030000 dyn.cannon_calc_drag

Job Frame 4

- **Queue is identical to 0.010**

Job Frames (Fifth)

```
...
0.010000 dyn_integloop.integ_sched.integrate
0.010000 dyn.cannon_deriv
0.010000 dyn.cannon_integ
0.010000 dyn.cannon_deriv
0.010000 dyn.cannon_integ

0.015000 dyn.cannon_calc_drag

0.020000 dyn_integloop.integ_sched.integrate
0.020000 dyn.cannon_deriv
0.020000 dyn.cannon_integ
0.020000 dyn.cannon_deriv
0.020000 dyn.cannon_integ

0.030000 dyn_integloop.integ_sched.integrate
0.030000 dyn.cannon_deriv
0.030000 dyn.cannon_integ
0.030000 dyn.cannon_deriv
0.030000 dyn.cannon_integ
0.030000 dyn.cannon_calc_drag
```

Job Frame 5

- Here we hit the LCM of 10 & 15, so all jobs are called and deriv-integ are called first

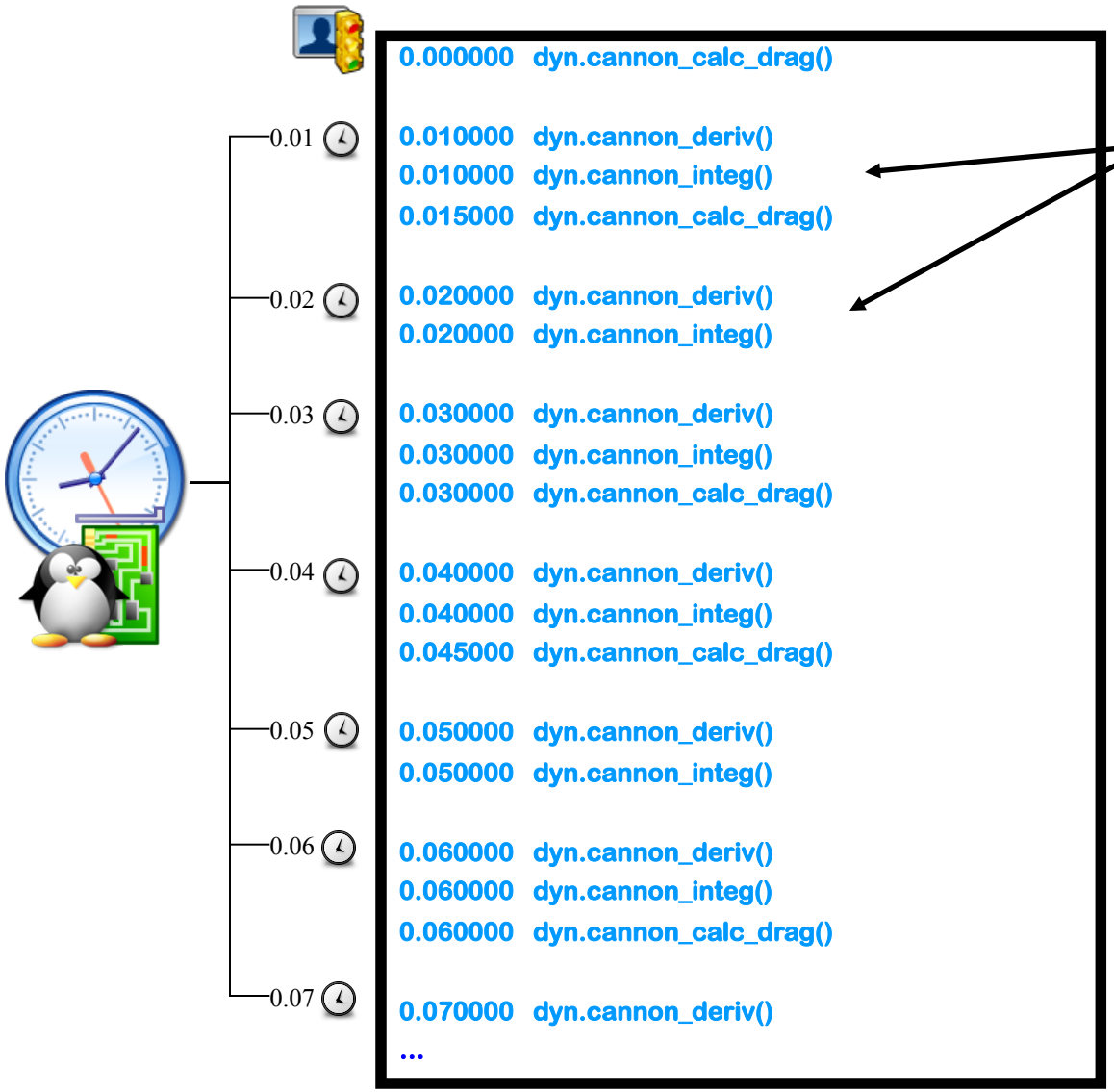
Job Frames (Notes On Time Stamp)

Various init jobs above

```
0.000000 dyn.cannon_deriv
0.000000 trick_ip.ip.process_event
0.000000 dyn.cannon_calc_drag
0.000000 trick_vs.vs.copy_data
0.000000 trick_sys.sched.advance_sim_time

0.010000 dyn_integloop.integ_sched.integrate
0.010000 dyn.cannon_deriv
0.010000 dyn.cannon_integ
0.010000 dyn.cannon_deriv
0.010000 dyn.cannon_integ

0.015000 dyn.cannon_calc_drag

0.020000 dyn_integloop.integ_sched.integrate
0.020000 dyn.cannon_deriv
0.020000 dyn.cannon_integ
0.020000 dyn.cannon_deriv
0.020000 dyn.cannon_integ

0.030000 dyn_integloop.integ_sched.integrate
0.030000 dyn.cannon_deriv
0.030000 dyn.cannon_integ
0.030000 dyn.cannon_deriv
0.030000 dyn.cannon_integ
0.030000 dyn.cannon_calc_drag
```

- **The frequency of a job tells Trick when to run it, NOT how long to run it**
- **The stamps are simulation time and have no relation to a wall clock**
- **"Real-time" can only occur (or make any sense definition-wise) if we decide to synchronize the simulation time with external real-time source**

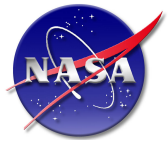
Real-Time Frame



This echo assumes single-step integration

It also assumes a real-time frame of 0.01

- **The real-time frame is the frequency at which Trick synchronizes with the wall clock**
- **The wall clock is either the system clock, interval timer signal or user defined clock**
- **All job frames within a RT frame run back to back**
- **Once all job frames are complete within each RT frame, a check is done on progress. If ahead of schedule, a wait is performed. If behind, the next software frame is immediately executed in an attempt to "catch up".**



Job Offset

- **Job Offset syntax**

```
(<frequency>, <offset>, <job class>) cannon_calc_drag( &cannon);
```

- **What if we changed the offset to cannon_calc_drag to 0.005**

```
(0.015, 0.005, "scheduled") cannon_calc_drag( &cannon);
```

Job Offsets

```
...
0.00000   dyn.cannon_deriv()

0.00500   dyn.cannon_calc_drag()

0.01000   dyn.cannon_deriv()
0.01000   dyn.cannon_integ()
0.01000   dyn.cannon_deriv()
0.01000   dyn.cannon_integ()

0.02000   dyn.cannon_deriv()
0.02000   dyn.cannon_integ()
0.02000   dyn.cannon_deriv()
0.02000   dyn.cannon_integ()
0.02000   dyn.cannon_calc_drag()

0.03000   dyn.cannon_deriv()
0.03000   dyn.cannon_integ()
0.03000   dyn.cannon_deriv()
0.03000   dyn.cannon_integ()

0.03500   dyn.cannon_calc_drag()
```

cannon_calc_drag is offset by 0.005 seconds

Job Phasing

- **Phasing**
  - Allows a user to reorder the jobs of the same class.
  - Originally used to reorder initialization jobs, but now extends to all job classes
  - Phase number ranges from 0 to 65534, default phase is 60000

```
P20 (0.015, "scheduled") cannon_calc_drag(…);
P10 (0.015, "scheduled") run_me_first(…);
```

```
0.00000    dyn.cannon_deriv()
0.00000    dyn.run_me_first()
0.00000    dyn.cannon_calc_drag()

0.01500    dyn.run_me_first()
0.01500    dyn.cannon_calc_drag()

0.03000    dyn.run_me_first()
0.03000    dyn.cannon_calc_drag()
```

Threading

Threading in Trick 10 works differently than in Trick 7.

```
C1 (1.0, "scheduled") slow_job(…);
```

C# identifies the thread in which a **scheduled** job runs.

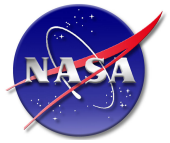
The **process type** of each thread may be set in the input file:

```
trick.exec_set_thread_process_type(1, trick.PROCESS_TYPE_ASYNC_CHILD)
```

There are three process types:

- **PROCESS_TYPE_SCHEDULED -** (the default) the child thread synchronizes with the main simulation thread at every time step.
- **PROCESS_TYPE_ASYNC_CHILD -** the thread does not synchronize with the main thread unless it ends.
- **PROCESS_TYPE_AMF_CHILD** specifies that the child thread periodically synchronizes with the main thread as specified in the input file:

```
trick.exec_set_thread_amf_cycle_time(1, 10.0)
```



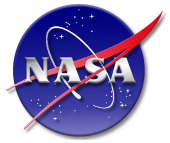
Collect Mechanism

- **What?**
  - Mechanism that allows a developer to group multiple parameters from various models, and of the same type, into a single parameter. Useful when the variable names or number of variables being passed may change.
  - Force summation is the most used application of the collect mechanism

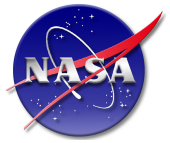
```
collect obj.struct.vars = { obj.struct1.variable_b , obj.struct2.var_c } ;
```

- **Where?**
  - The collect statement is in the S_define. A job is designated to process the values "collected".

- **How?**
  - The list of parameters in the collect statement is turned into an array of values, and a pointer is assigned to that array..



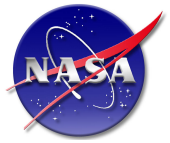
Input Processor



Input Processor

- **Objective**
  - Describe how the input processor works with a Trick simulation.
  - Show examples of how the input file can affect the simulation

- **Prerequisites**
  - None



Input Processor

The input processor is a Python intrepreter that "knows" about your simulation's sim objects.

A simulation input file is simply a Python script. It is typically named **input.py**.

Typical invocation of a simulation is :

```
% ./S_main*.exe RUN_test/input.py
```

The input processor runs input.py to completion before user's initialization jobs are run.

There is an online Python Tutorial at: **http://docs.python.org/tutorial/**

Input Processor

Assign Values :

```
dyn.cannon.pos[0] = 0.0
dyn.cannon.pos[1] = 0.0
dyn.cannon.acc[0] = 0.0
dyn.cannon.acc[1] = -9.81
dyn.cannon.init_angle = trick.attach_units("d",30.0)
dyn.cannon.init_speed  = 50.0
```

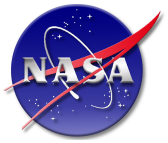
Units Conversion

Tell the sim to stop after 300 seconds :

```
trick.exec_set_terminate_time(300.0)
```

An alternate way of stopping:

```
trick.stop(300.0)
```



How to write a :

Pre-initialization checkpoint :

```
trick.checkpoint_pre_init(True)
```

Post-initialization checkpoint :

```
trick.checkpoint_post_init(True)
```

Checkpoint at 120 seconds:

```
trick.checkpoint(120.0)
```

Post-Run Checkpoint :

```
trick.checkpoint_end(True)
```

How to provide a :

Sim Control Panel :

```
trick.sim_control_panel_set_enabled(1)
```

Trick View:

```
trick.trick_view_set_enabled(1)
```

Stripchart:

```
trick_vs.stripchart.set_input_file("cannon.sc")
trick.stripchart_set_enabled(1)
```

Monte Carlo Monitor :

```
trick.monte_monitor_set_enabled(1)
```

Malfunctions Panel:

```
trick.malfunctions_trick_view_set_enabled(1)
```

How to :

Set the job rate:

```
trick.set_job_cycle( char* job_name, double in_cycle)
```

Turn a job on/off

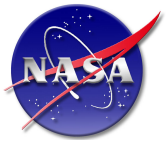
```
trick.exec_set_job_onoff(char * job_name , int on_off)
```

Turn a sim object on/off

```
trick.exec_set_sim_object_onoff( char* sim_obj_name, int on_off)
```

Turn echo jobs on/off.

```
trick.echo_jobs_on()
```



Input Processor – Creating an Integrator

Create a Runge_Kutta_2, four element state integrator whose default integration rate is 0.01 seconds :

```
integ_loop_object.getIntegrator( trick.Runge_Kutta_2, 4)
```

Create a Runge_Kutta_4, eight element state integrator whose default integration rate is 0.01 seconds :

```
integ_loop_object.getIntegrator( trick.Runge_Kutta_4,8)
```

Add_read

Executing Python code at a specific time using `add_read()`:

```
% cd $HOME/trick_sims/SIM_cannon_L4
% vi RUN_grav/input.py
```

Uncomment the code below

```
#add_read example
read = 2.3
trick.add_read(read,"""print "hello, the sim time is: " , trick.exec_get_sim_time()""")
read = 3.4
trick.add_read(read,"""print "howdy, the sim time is: " , trick.exec_get_sim_time()""")
```

Comment-out the line below and save

```
#trick.echo_jobs_on()
```

Re-run the sim.

```
% ./S_main_${TRICK_HOST_CPU}.exe RUN_grav/input.py
```

On your terminal, you should see:

```
hello, the sim time is:  2.3
howdy, the sim time is:  3.4
```

Input Processor – Events

Now, let's add an event :

```
% cd $HOME/trick_sims/SIM_cannon_L4
% vi RUN_grav/input.py <uncomment the code below and save>
```

```
# --- Event Example ---
ceiling = 25.0
event_1 = trick.new_event("event_1")
event_1.set_cycle(0.01)
event_1.condition(0, """dyn.cannon.pos[1] > ceiling""")
event_1.action(0, """print 'Hit the ceiling at', trick.exec_get_sim_time(),\
'seconds.'""")
trick.add_event(event_1)
event_1.activate()
```

Re-run the sim.

```
% ./S_main_${TRICK_HOST_CPU}.exe RUN_grav/input.py
```

You should see: `Hit the ceiling at 1.37 seconds`

Re-activating an Event

Like a mousetrap, once an event is triggered, it is no longer active. To Reset the event, use: *`event`*`.activate()` in the *`event`*`.action` string:

```
# --- Event Example #2 ---
event_2 = trick.new_event("event_2")
event_2.set_cycle(0.01)
event_2.condition(0, """((dyn.cannon.pos[1] > ceiling) and\
(dyn.cannon.pos[1] - float(dyn.cannon.vel[1] * 0.01)) < ceiling) or\
((dyn.cannon.pos[1] < ceiling) and\
(dyn.cannon.pos[1] - float(dyn.cannon.vel[1] * 0.01)) > ceiling)""")
event_2.action(0, """print 'Hit the ceiling at', trick.exec_get_sim_time(),\
'seconds.', event_2.activate() """)
trick.add_event(event_2)
event_2.activate()
```

Reactivating an event

Now let's stripchart the cannonball trajectory:

A stripchart configuration file specifies what our stripchart will contain (note, parameters with the equal sign are optional):

FILE

cannon.sc :

```
Stripchart:
   title = "Cannon Trajectory"
   geometry = 800x800+300+0
   x_min = 0.0
   x_max = 250.0
   y_min = 0.0
   y_max = 40.0
   x_variable = dyn.cannon.pos[0]
   dyn.cannon.pos[1]
```

```
% cd $HOME/trick_sims/SIM_cannon_L4
% vi RUN_grav/input.py <uncomment the code below and save>
```

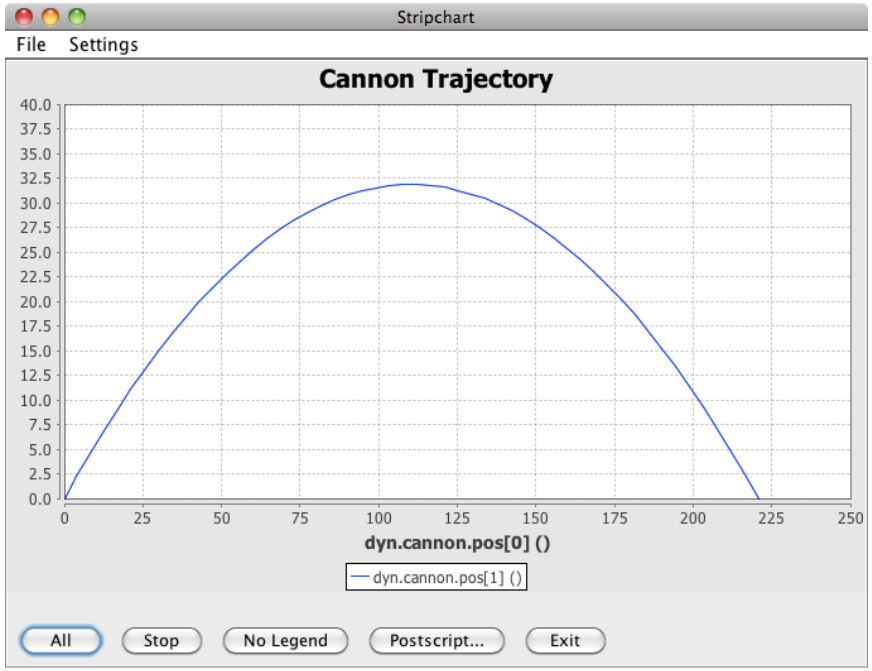
```
trick_vs.stripchart.set_input_file("cannon.sc")
trick.stripchart_set_enabled(1)
```

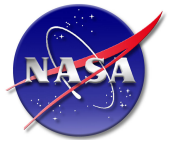
Input Processor – Events Example

Re-run the sim.

```
% ./S_main_${TRICK_HOST_CPU}.exe RUN_grav/input.py
```

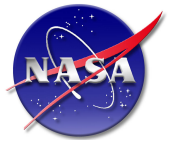
You should see a stripchart similar to this, produced during the simulation:





Trick DP

(DP = Data Products)



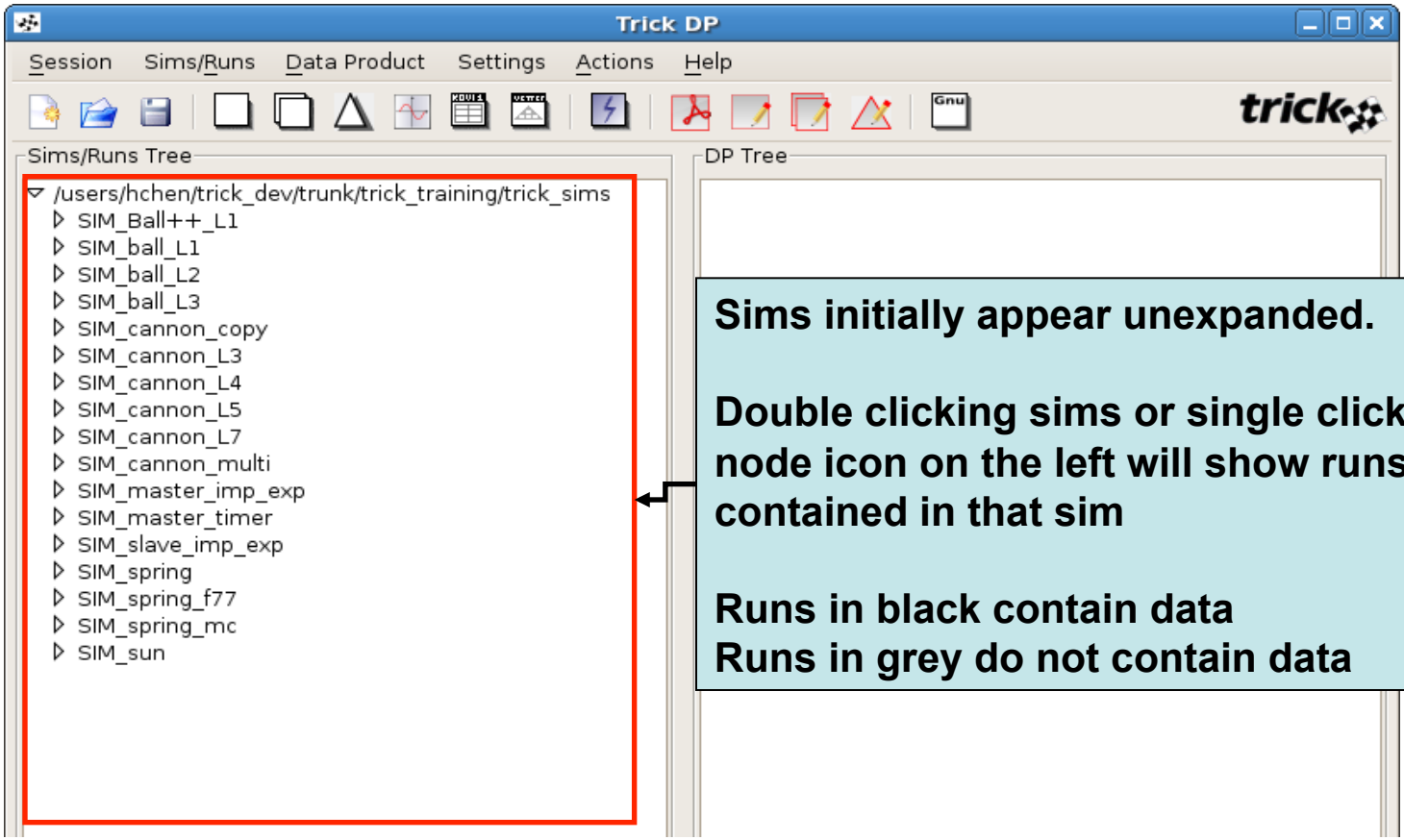
Trick DP Advanced Topics

- **Objective**
  - **Create new pages/plots**
  - **Create and save DP "sessions"**

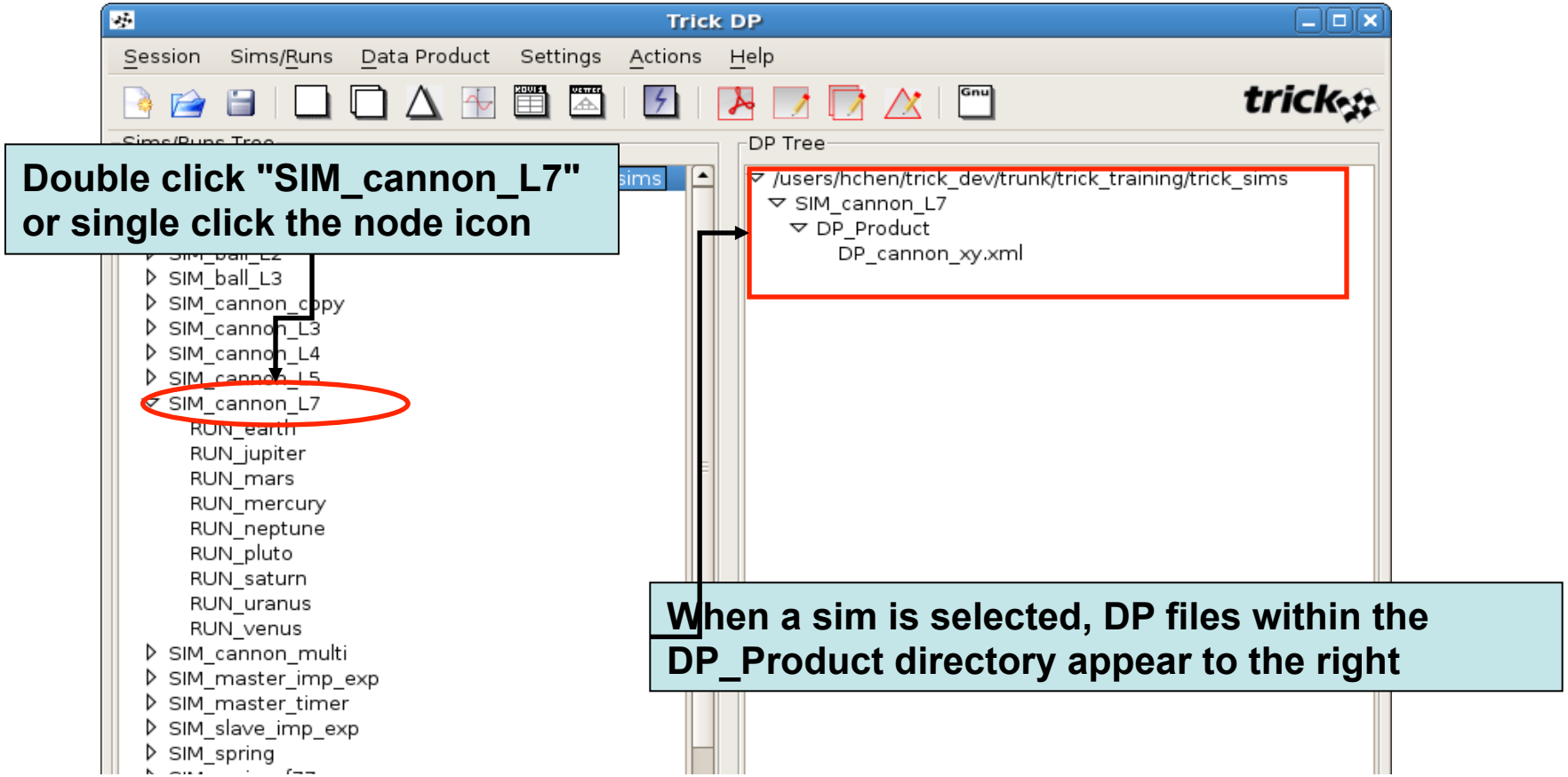
- **Prerequisites**
  - **Data recorded when simulation run**

- **Start trick_dp**

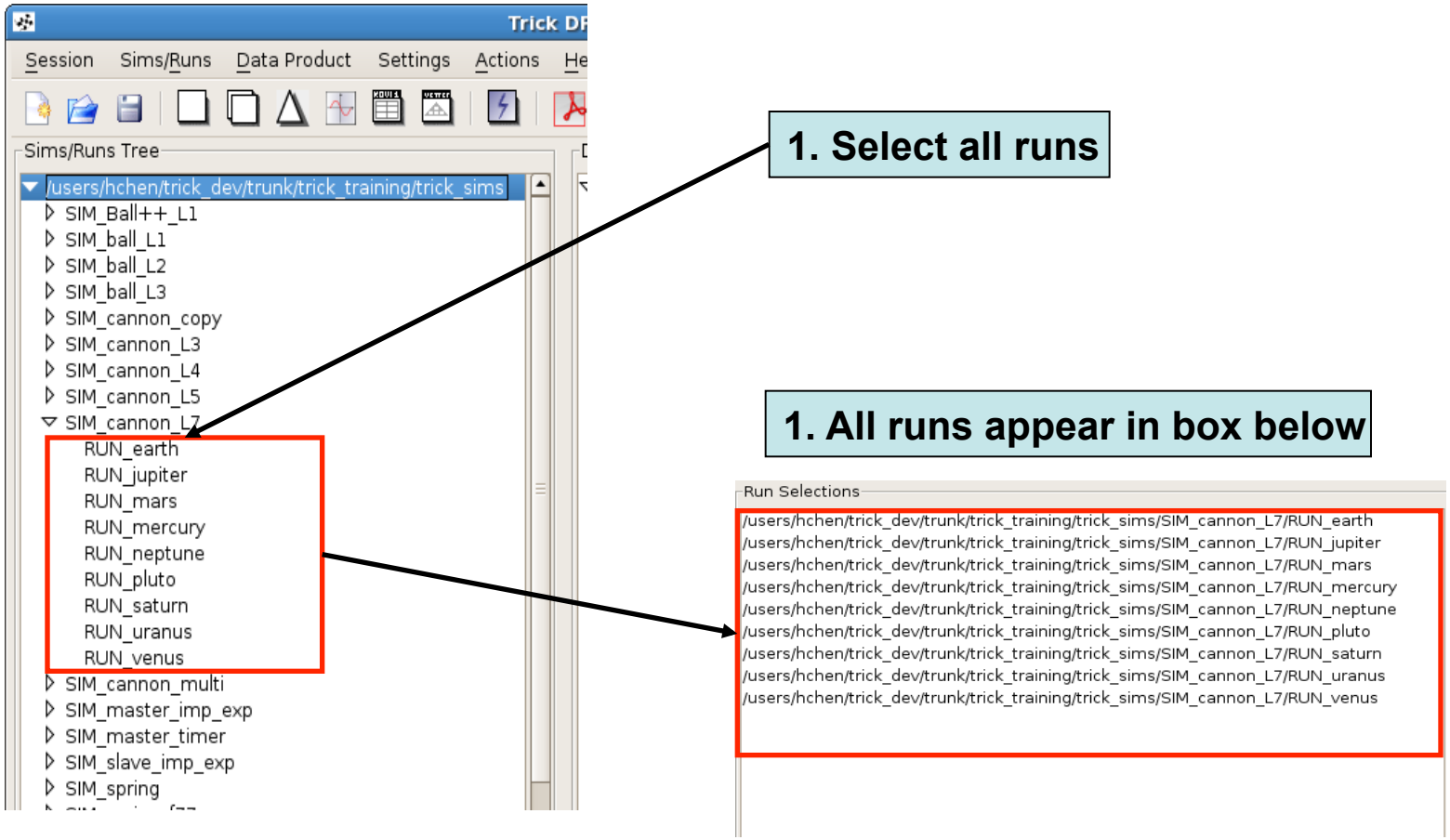
```
% cd $HOME/trick_sims/SIM_cannon_L7
% trick_dp &
```



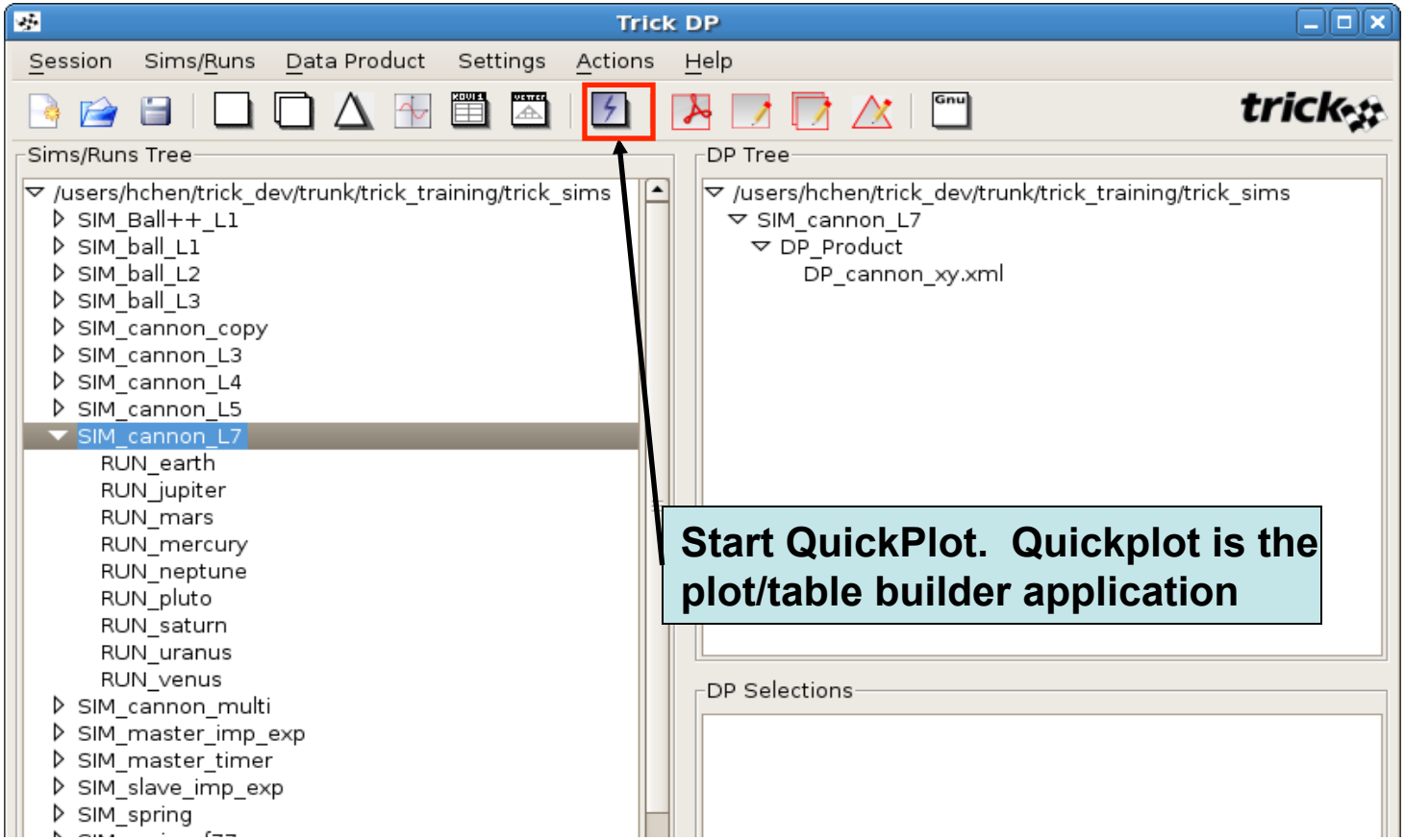
- **Open SIM_cannon_L7**



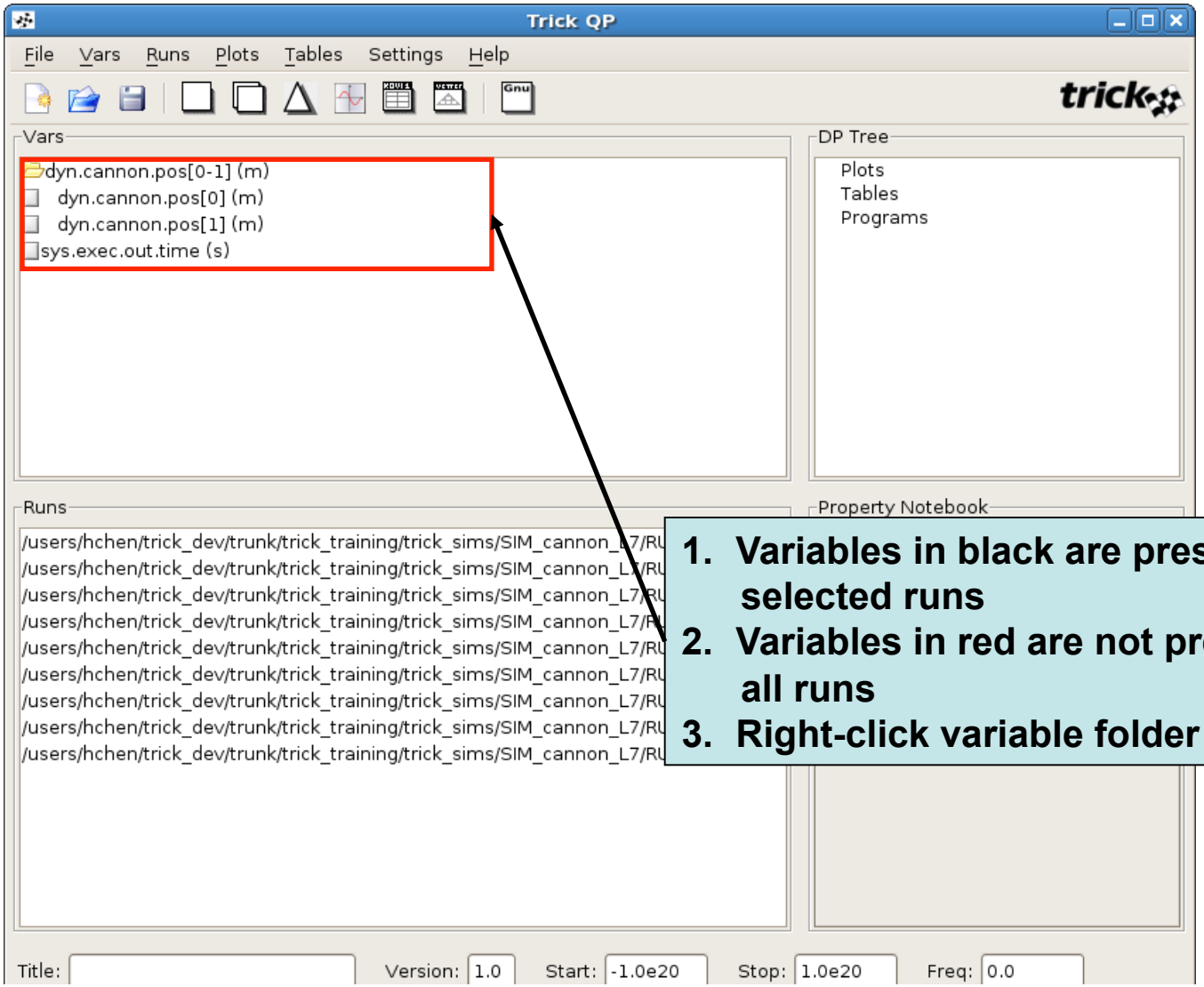
- **Select all runs in the SIM by double clicking each one or by right-clicking the SIM folder and clicking "Add run(s)"**



- **Start Quickplot**

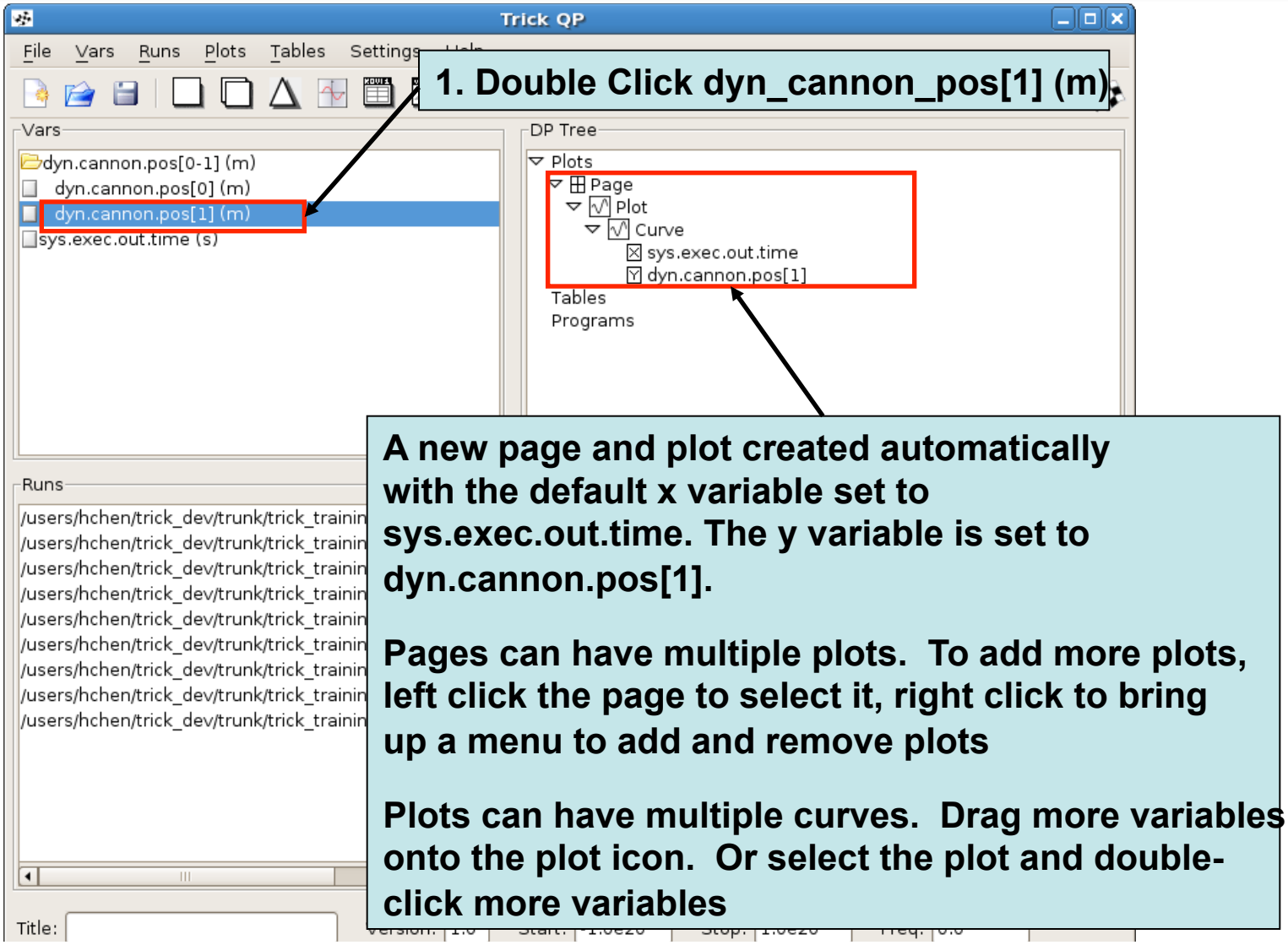


Creating a DP File

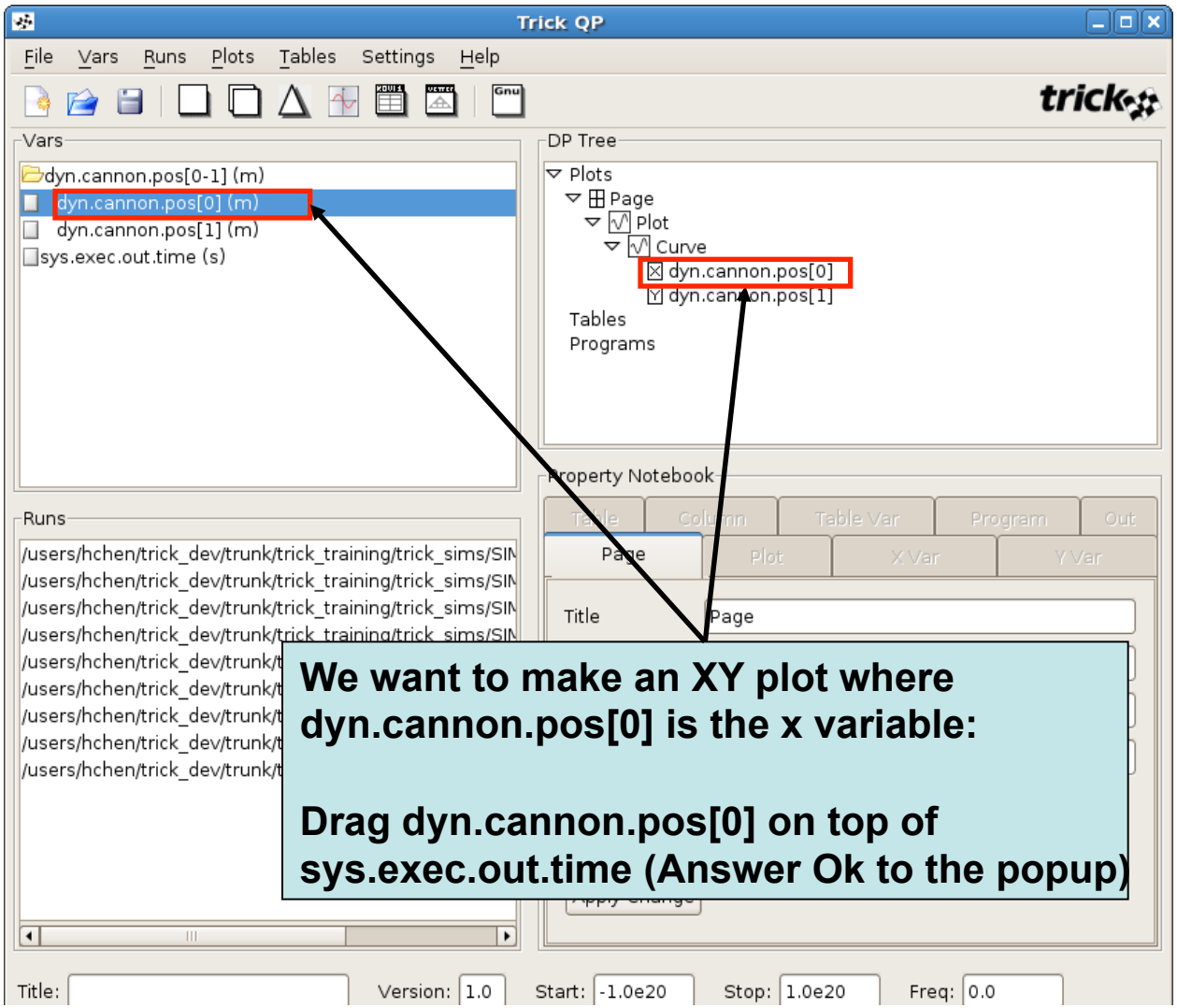


1. **Variables in black are present in all selected runs**
2. **Variables in red are not present in all runs**
3. **Right-click variable folder to expand**

Creating a DP File

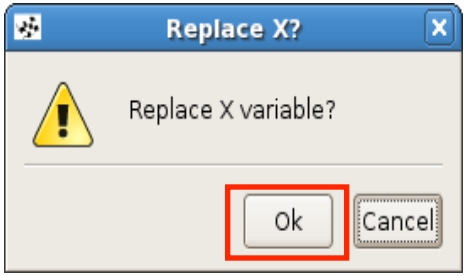


Creating a DP File

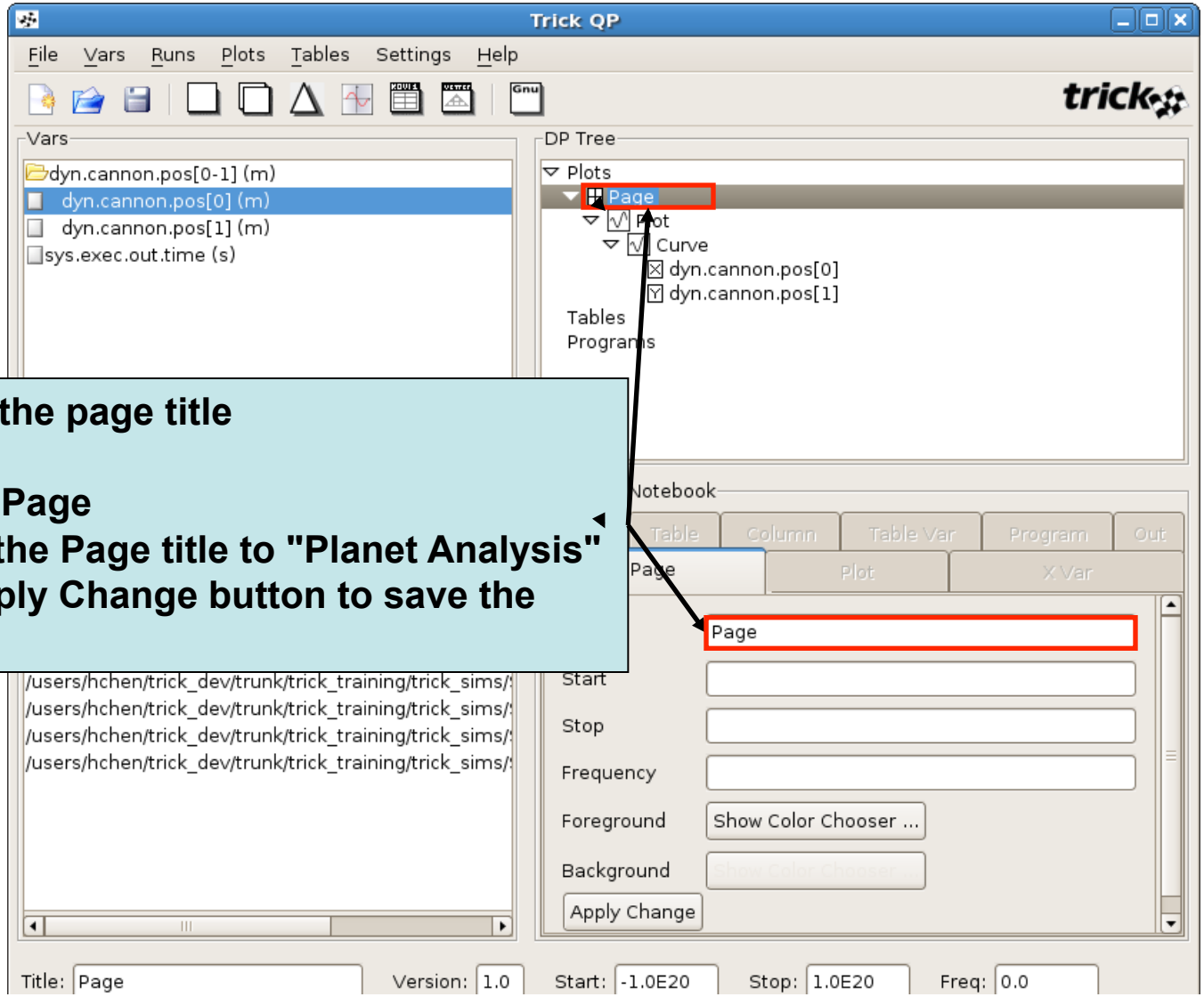


We want to make an XY plot where dyn.cannon.pos[0] is the x variable:

Drag dyn.cannon.pos[0] on top of sys.exec.out.time (Answer Ok to the popup)



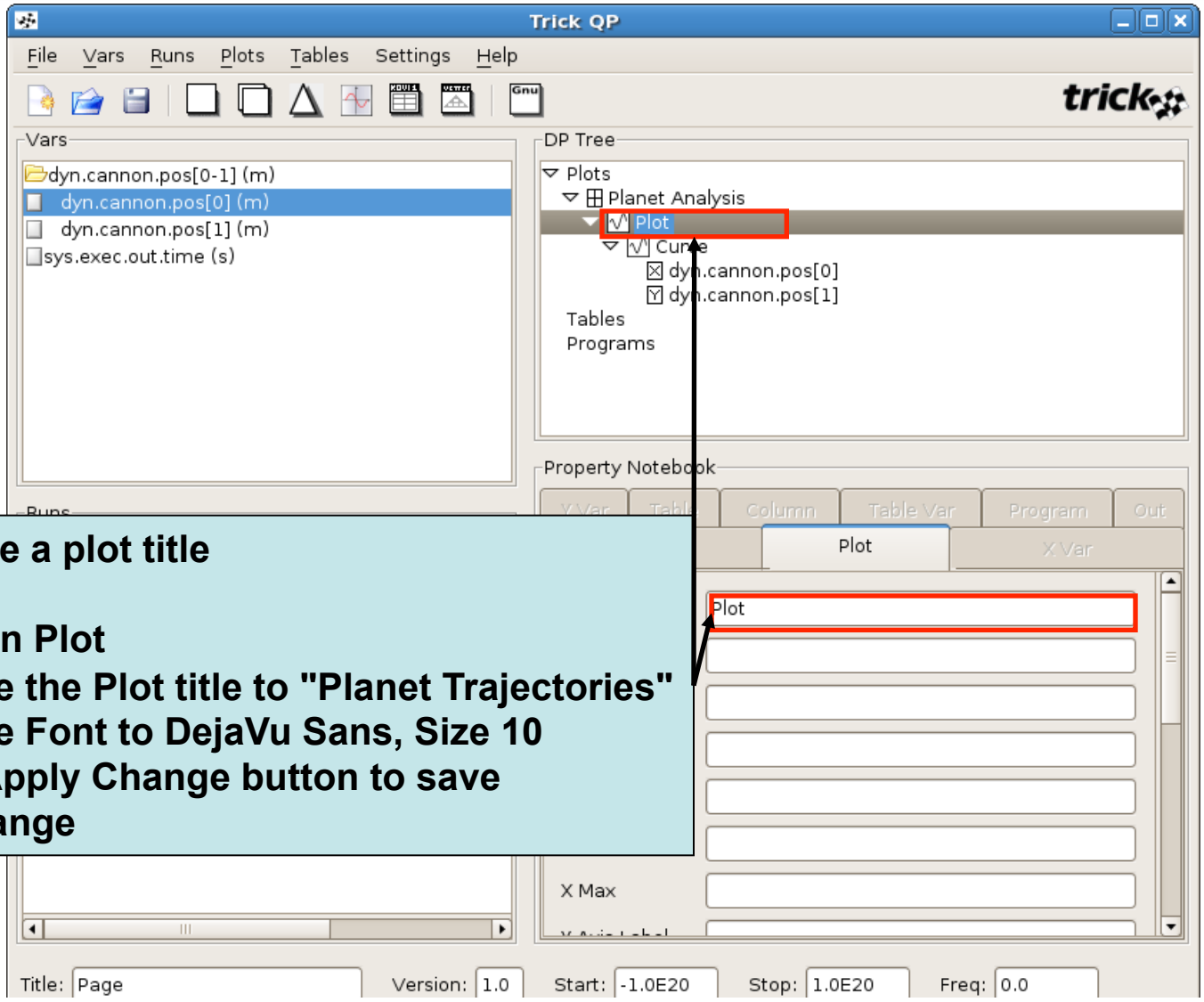
Creating a DP File



To change the page title

1. Click on Page
2. Change the Page title to "Planet Analysis"
3. Click Apply Change button to save the change

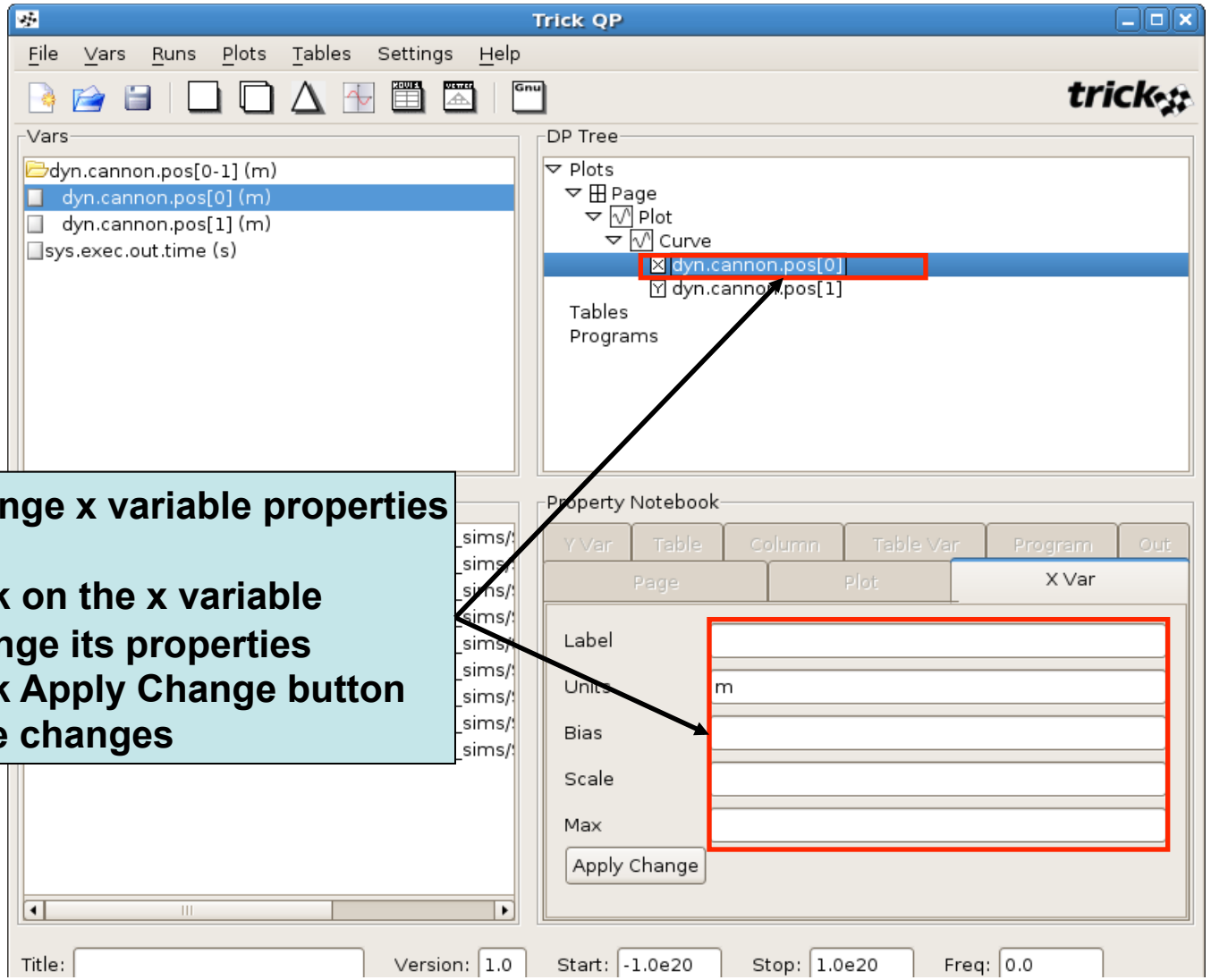
Creating a DP File



To change a plot title

1. **Click on Plot**
2. **Change the Plot title to "Planet Trajectories"**
3. **Change Font to DejaVu Sans, Size 10**
4. **Click Apply Change button to save the change**

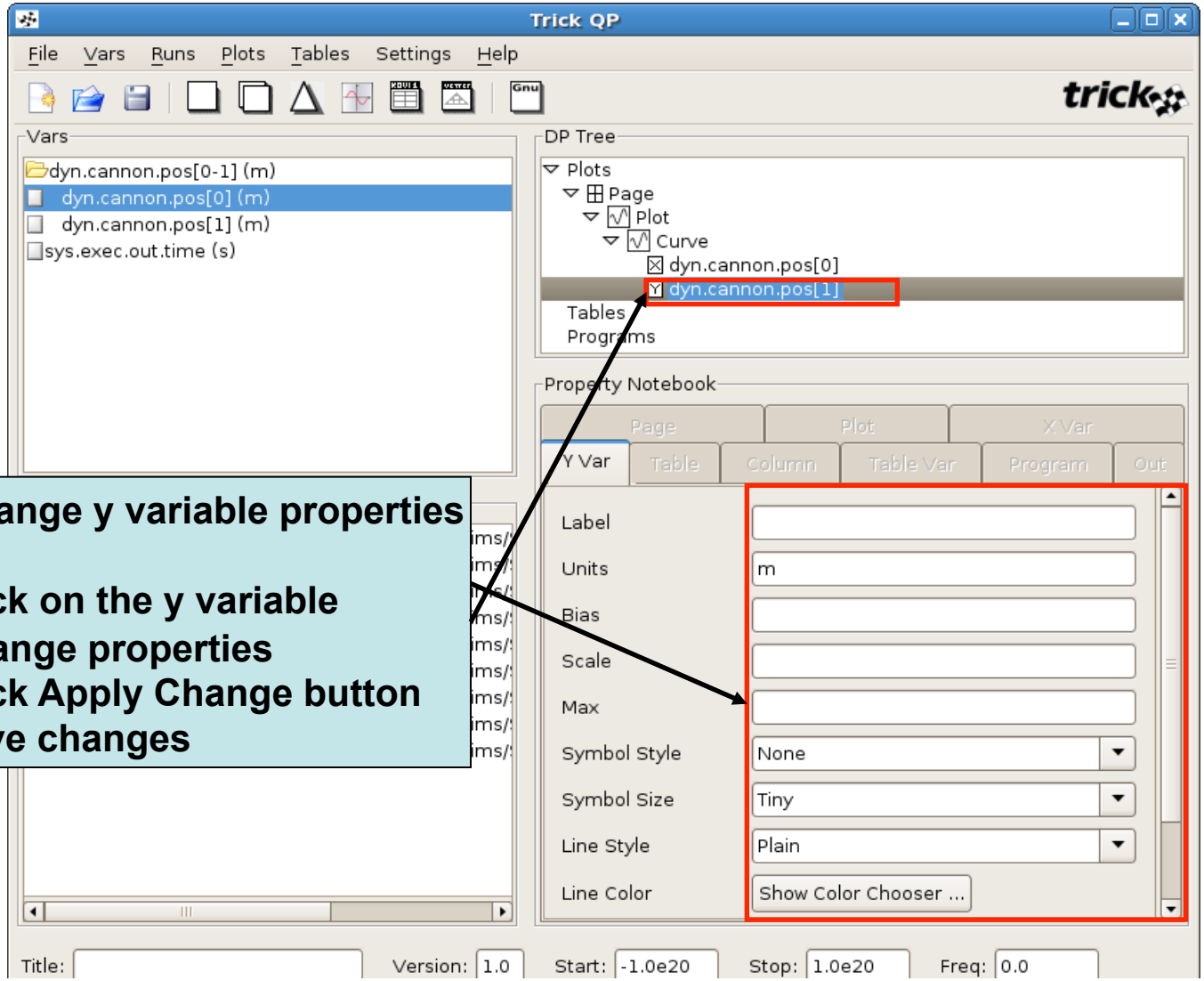
Creating a DP file



To change x variable properties

1. Click on the x variable
2. Change its properties
3. Click Apply Change button to save changes

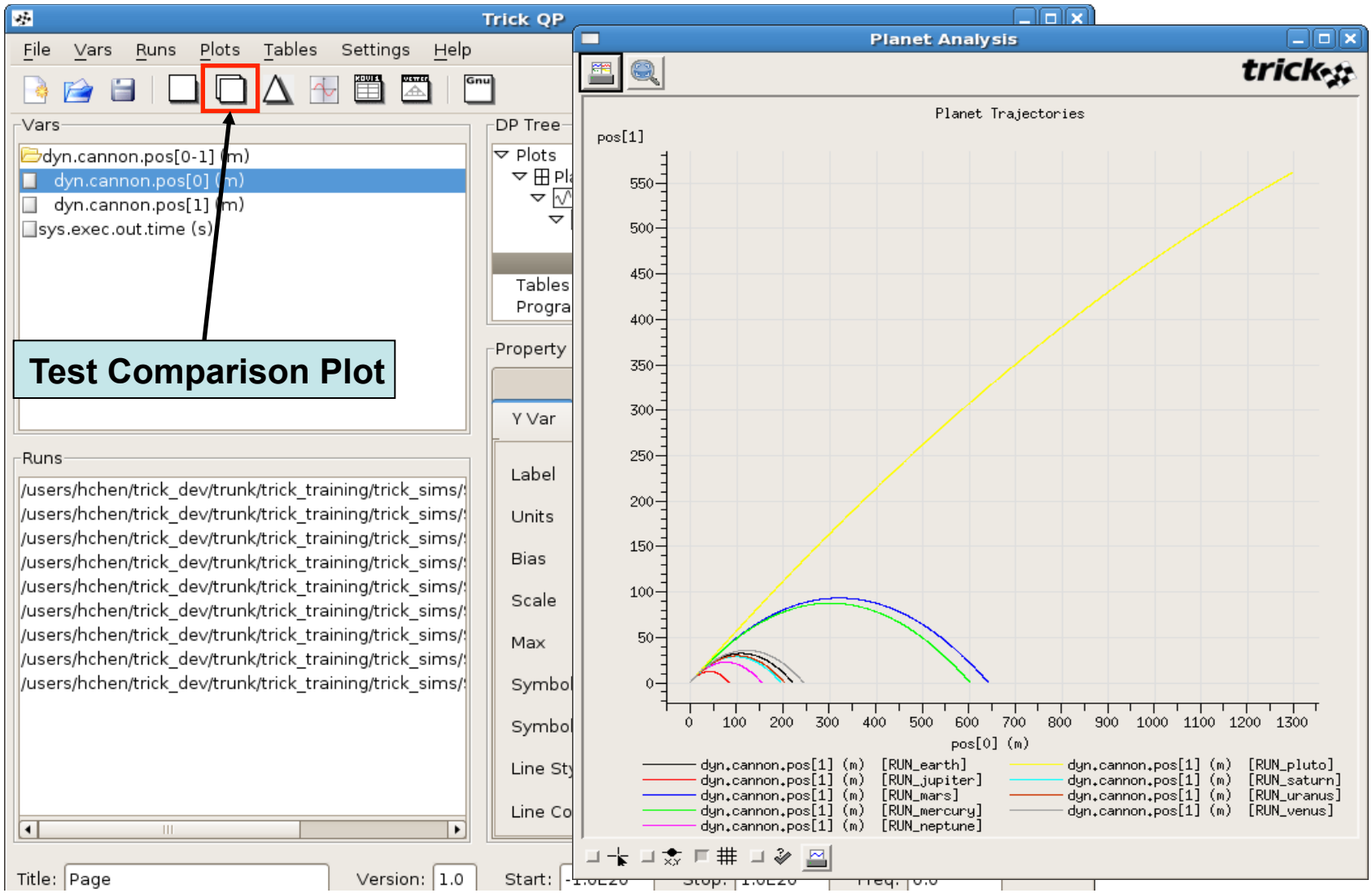
Creating a DP file



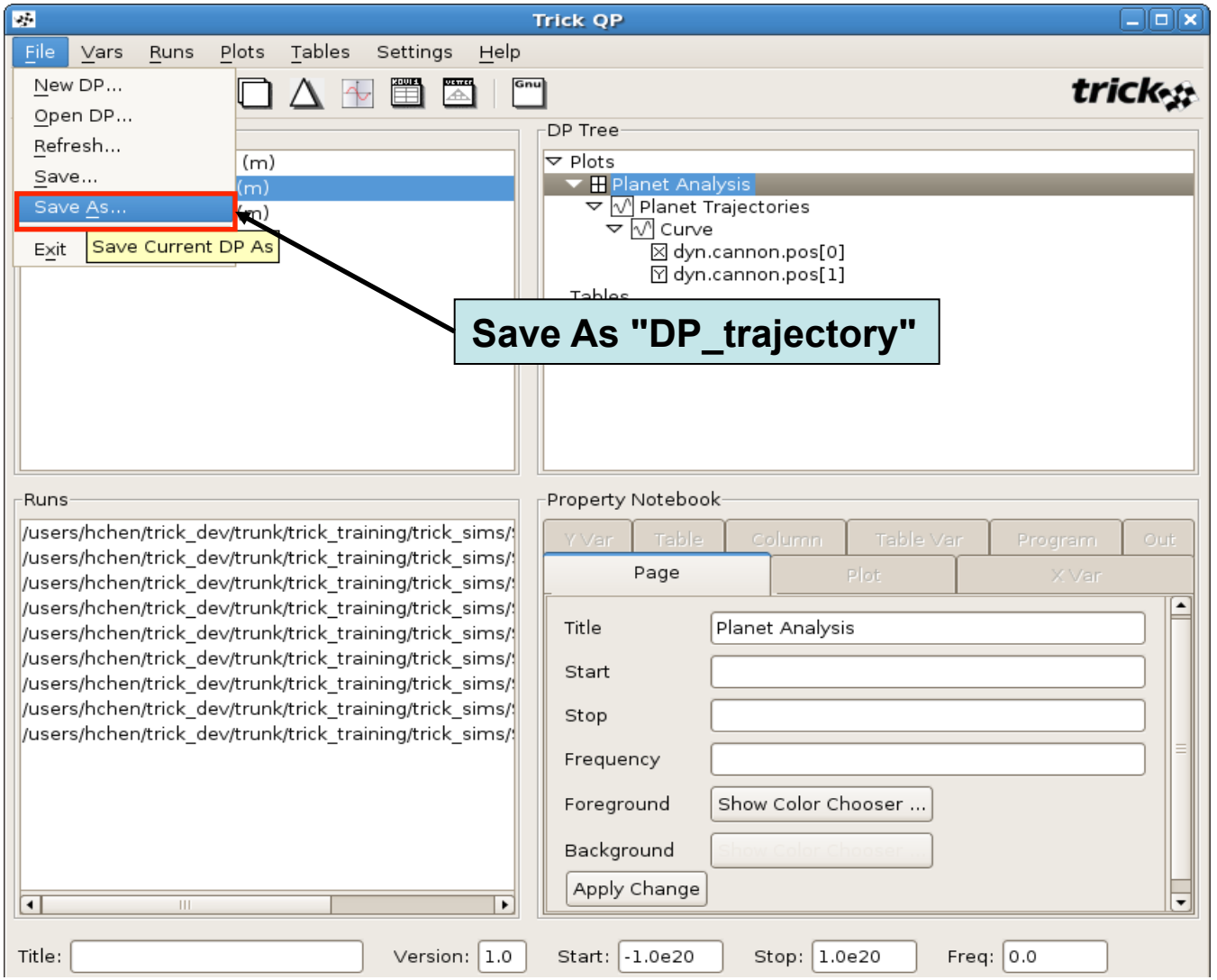
To change y variable properties

1. Click on the y variable
2. Change properties
3. Click Apply Change button to save changes

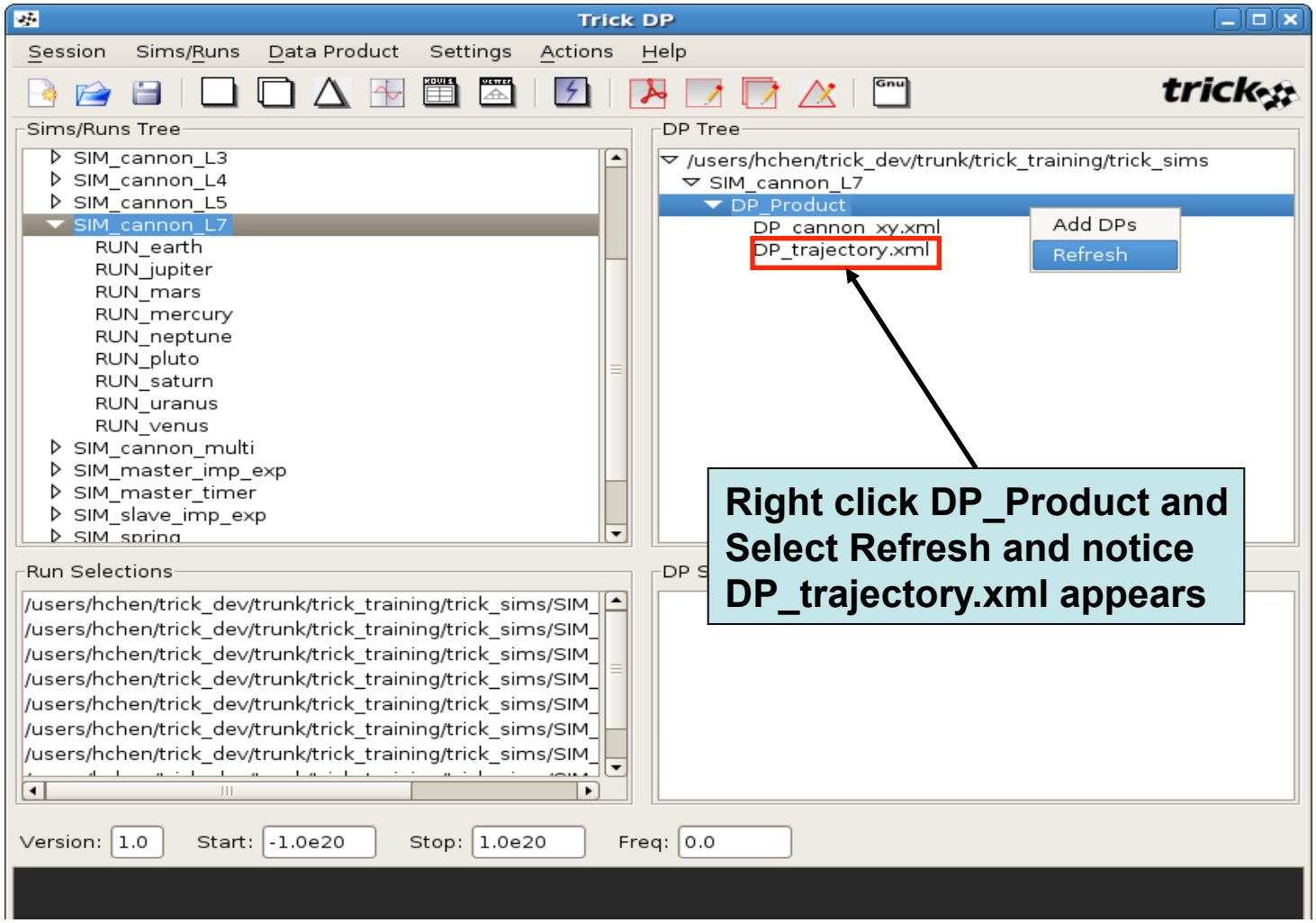
Creating a DP File



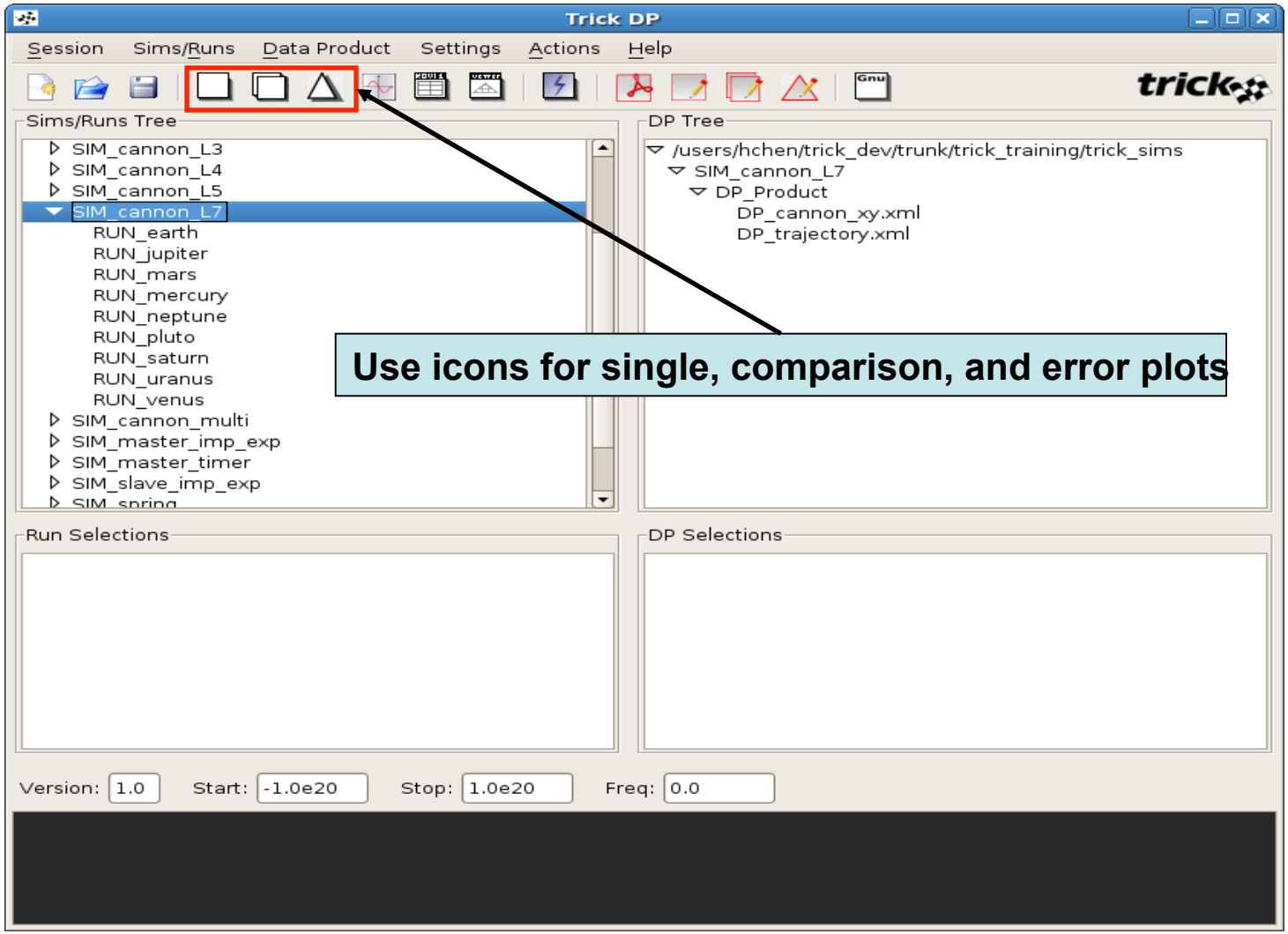
Creating a DP File



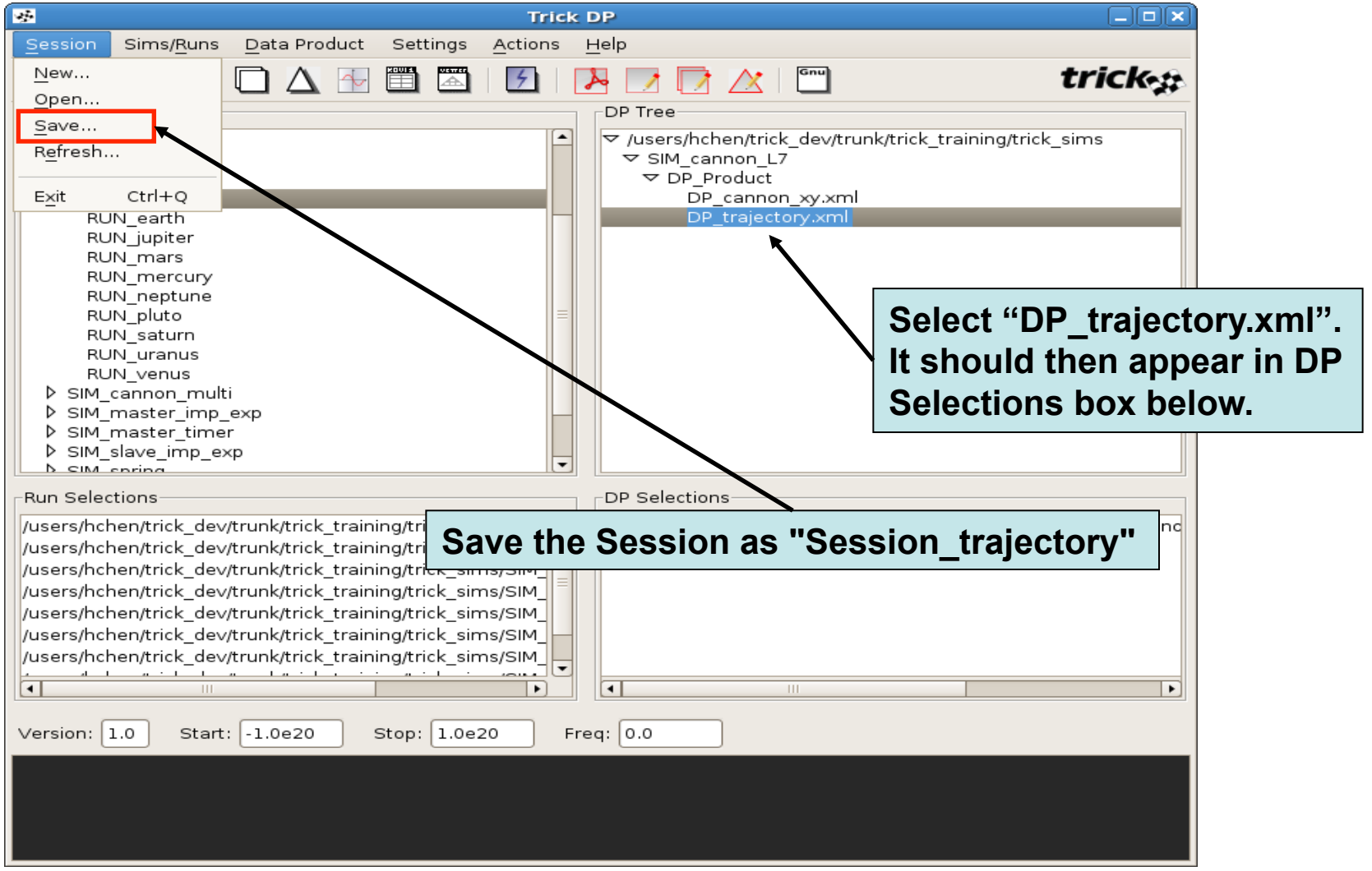
Creating a DP File



Plotting


Trick DP
Session
Sims/Runs
Data Product
Settings
Actions
Help
trick
Sims/Runs Tree
SIM_cannon_L3
SIM_cannon_L4
SIM_cannon_L5
SIM_cannon_L7
RUN_earth
RUN_jupiter
RUN_mars
RUN_mercury
RUN_neptune
RUN_pluto
RUN_saturn
RUN_uranus
RUN_venus
SIM_cannon_multi
SIM_master_imp_exp
SIM_master_timer
SIM_slave_imp_exp
SIM_spring
DP Tree
/users/hchen/trick_dev/trunk/trick_training/trick_sims
SIM_cannon_L7
DP_Product
DP_cannon_xy.xml
DP_trajectory.xml
Use icons for single, comparison, and error plots
Run Selections
DP Selections
Version: 1.0
Start: -1.0e20
Stop: 1.0e20
Freq: 0.0

Saving DP Session

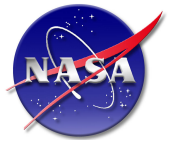


- **Plotting from the command line**

Plotting from the command line

(Session_trajectory.xml is saved by default in you SIM_ directory)

% fxplot Session_trajectory.xml (fxplot uses enhanced Fermi-lab X-widget)



Data Recording

- **How do you set up the sim data used by Data Products?**
  - **Create a data recording file using data recording editor (dre)**
  - **You need to be in the directory that contains the S_sie.resource file created by CP in order to launch dre successfully**

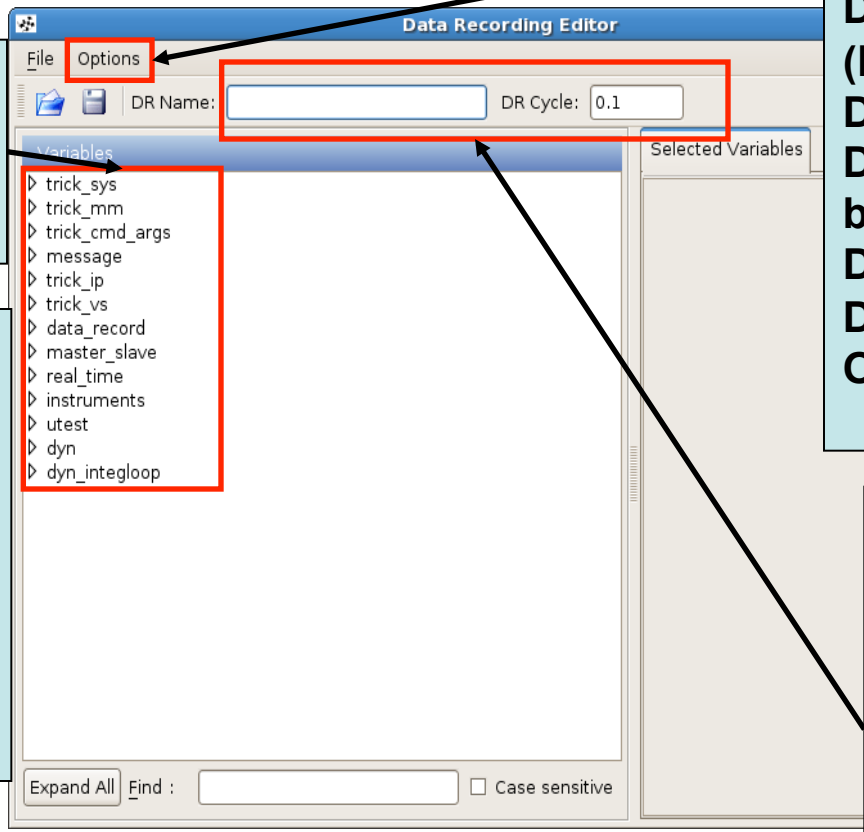
```
% cd $HOME/trick_sims/SIM_cannon_example   <if your not already in this directory>
% dre
```

Data Recording

Click on hierarchy in left window to find variables to record

Double click variables on left adds to the references in right Selected Variables window

Double click "dyn" or single click the node icon

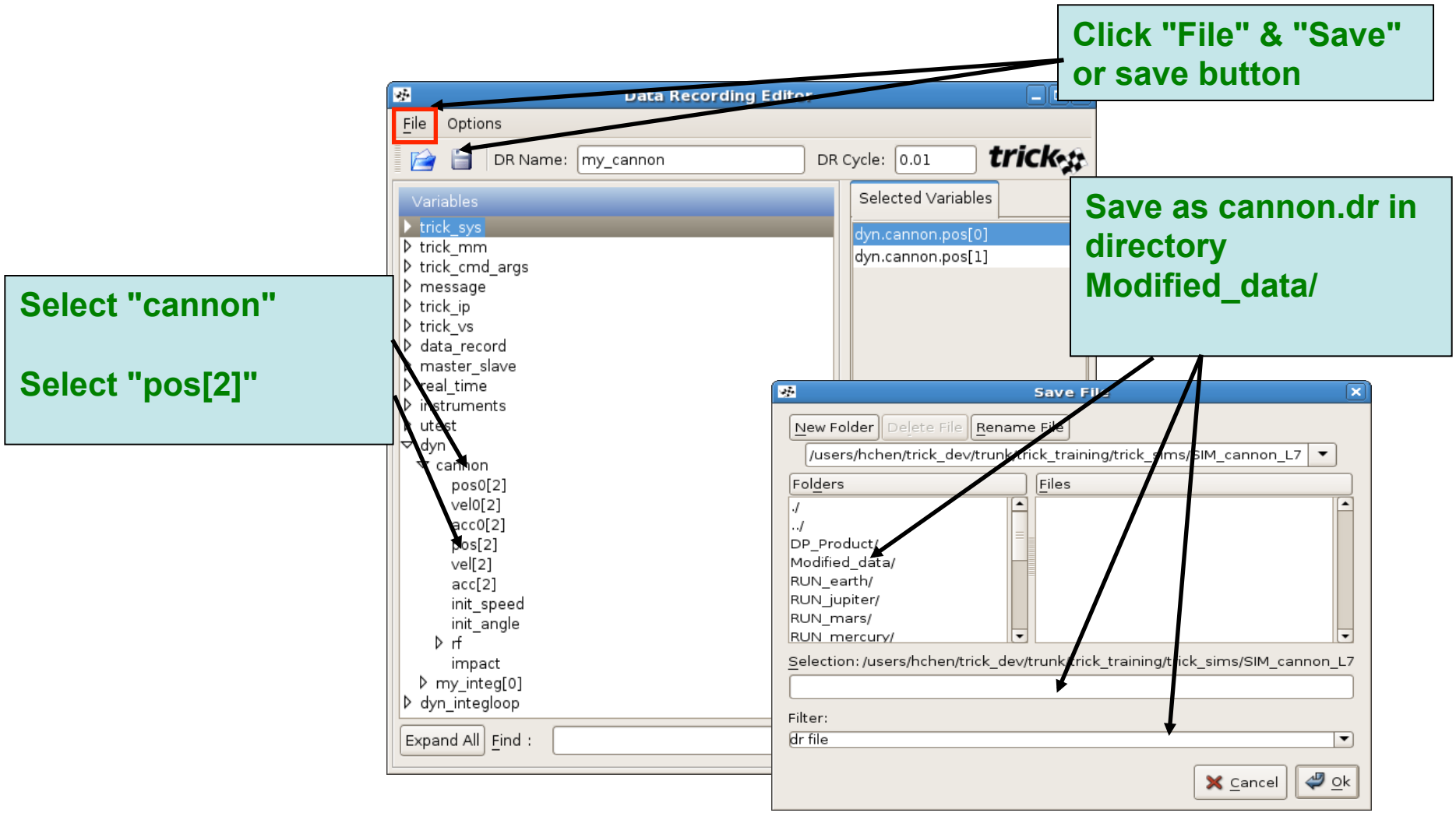


Set recording format (DR_Binary, Dr_ASCII, & DR_HDF5), frequency (DR_Always, DR_Changes, and DR_Step_Changes), buffering(DR_Buffer, DR_No_Buffer, DR_Ring_Buffer) under Options

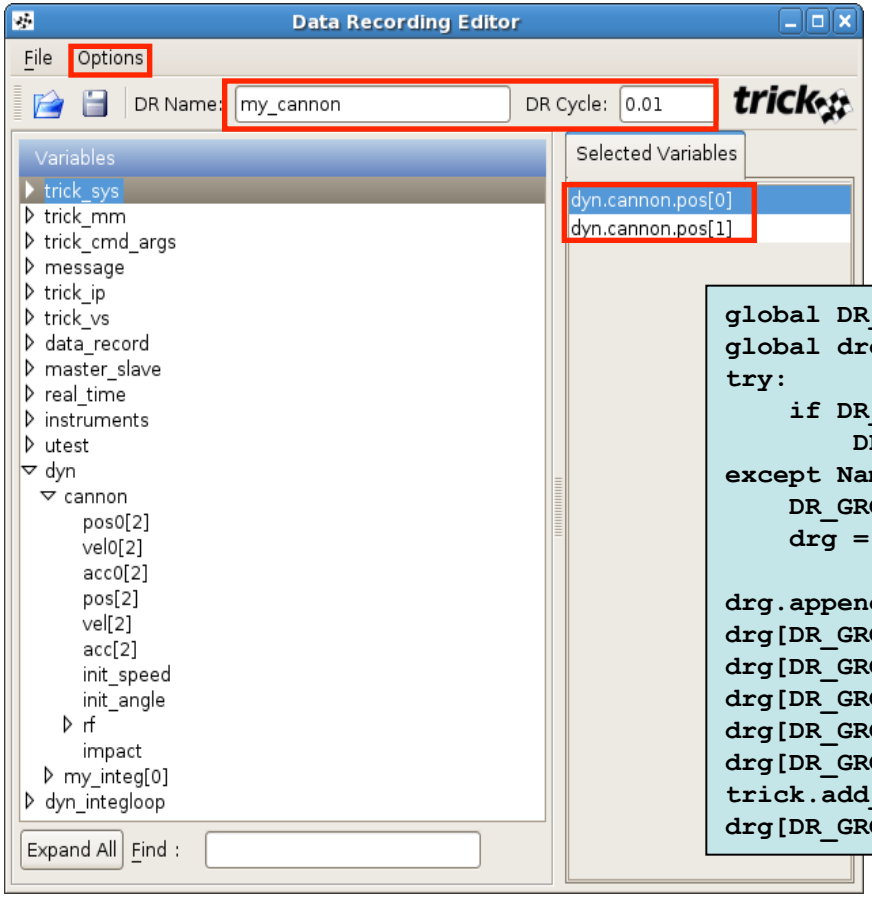
Give this recording group a name and the cycle time to record

In DR Name put "my_cannon"
Change DR Cycle to "0.01"

Data Recording

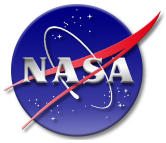


- **Data Recording auto-generated file**




```
global DR_GROUP_ID
global drg
try:
    if DR_GROUP_ID >= 0:
        DR_GROUP_ID += 1
except NameError:
    DR_GROUP_ID = 0
    drg = []

drg.append(trick.DRBinary("my_cannon"))
drg[DR_GROUP_ID].set_freq(trick.DR_Always)
drg[DR_GROUP_ID].set_cycle(0.01)
drg[DR_GROUP_ID].set_single_prec_only(False)
drg[DR_GROUP_ID].add_variable("dyn.cannon.pos[0]")
drg[DR_GROUP_ID].add_variable("dyn.cannon.pos[1]")
trick.add_data_record_group(drg[DR_GROUP_ID], trick.DR_Buffer)
drg[DR_GROUP_ID].enable()
```

[see Trick User's Guide § 7.8]



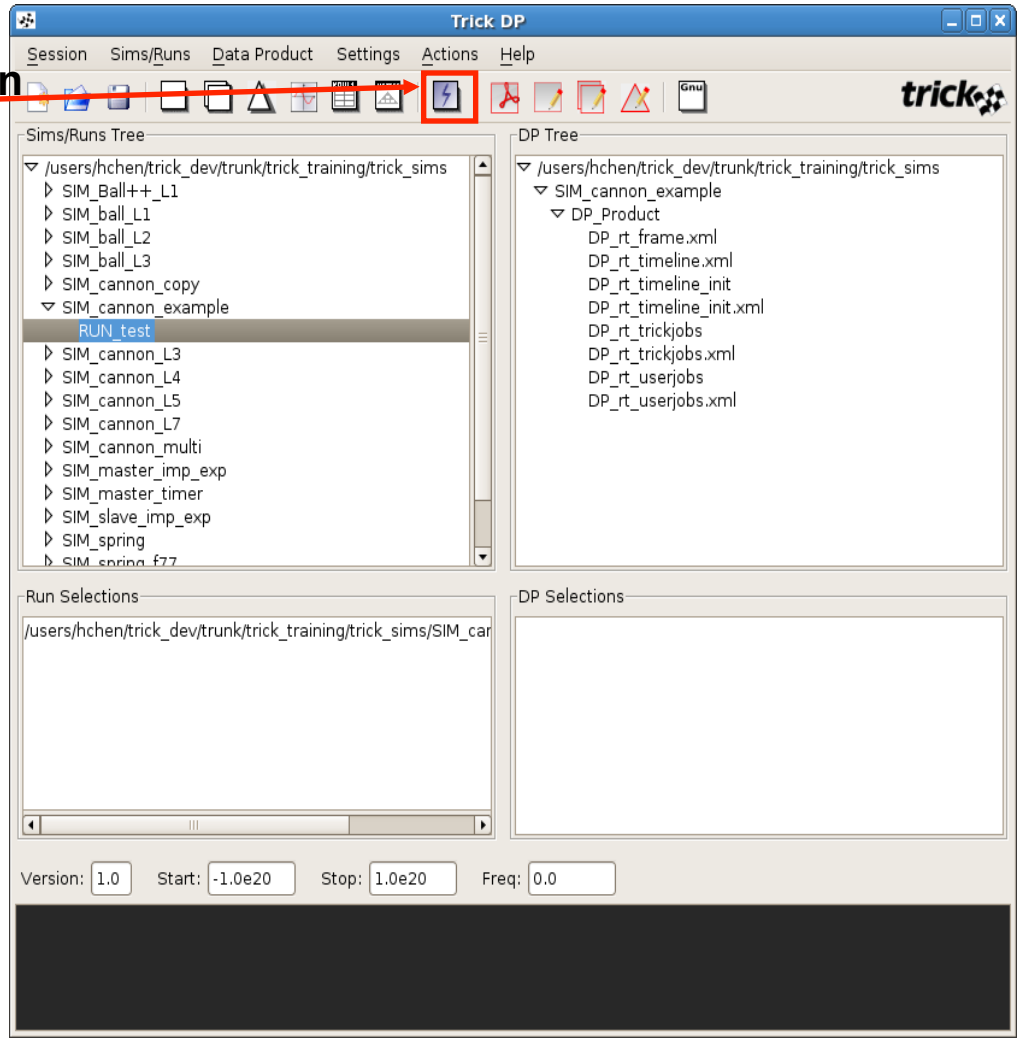
Data Recording

```
% vi RUN_test/input.py <edit as below and save>

execfile("Modified_data/cannon.dr")
execfile("Modified_data/realtime.py")

dyn_integloop.getIntegrator(trick.Runge_Kutta_4, 4)
my_event = trick.new_event("impact")
my_event.set_cycle(0.01)
my_event.condition(0,"""trick.exec_get_sim_time() > 1.0 and \
                      dyn.cannon.pos[1] <= 0.0""")
my_event.action(0,"""print 'impact time: %f X-position: %s Y-position: %s' \
                   %(trick.exec_get_sim_time(), dyn.cannon.pos[0], \
                   dyn.cannon.pos[1])""")
trick.add_event(my_event)
my_event.activate()

trick.stop(5.2)
```

```
% ./S_main*exe RUN_test/input.py &
% trick_dp &
```

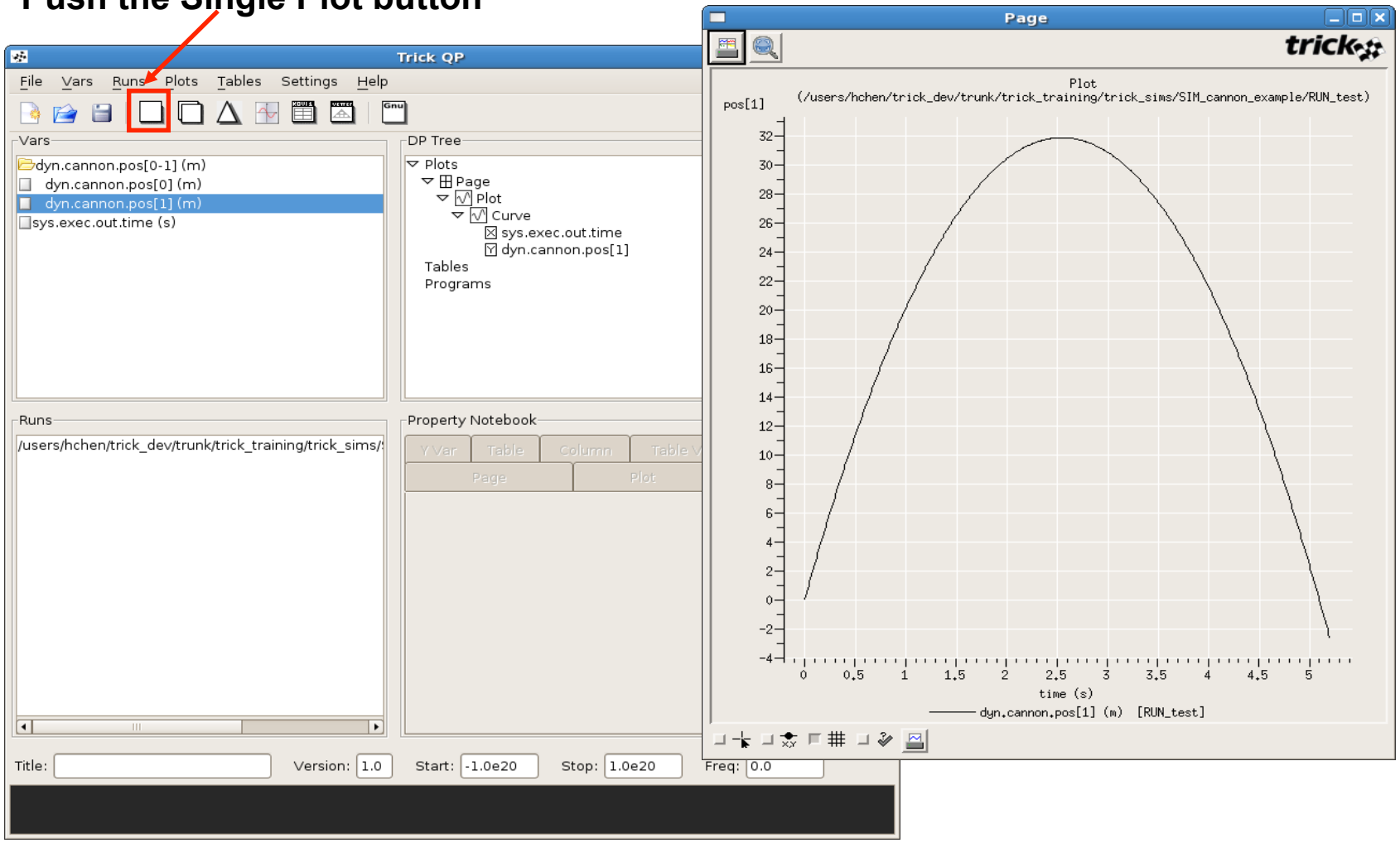
Data Recording

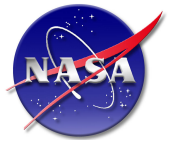
- Double Click SIM_cannon_example in the Sims/Runs window
- Double click RUN_test
- Push the Quick Plot button



Data Recording

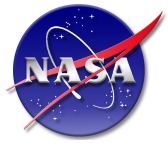
- **Double Click dyn.cannon.pos[1] (m) – note, right-click to expand**
- **Push the Single Plot button**





Data Recording Formats

- **When Trick sims log data they can use 3 recording formats**
  - **DRBinary format (the default) --> <filename>.trk**
  - **DRAscii --> <filename>.csv**
  - **DRHDF5, readable by Matlab --> <filename>.h5**
- **Logged data files are placed in the RUN directory**
- **Use DRAscii or DRHDF5 recording to export Trick data to other programs**
- **Trick data products can read data from**
  - **Trick native formats: Trick Binary, CSV, HDF5**



End of Day 1

- You have made it through the Basic Tutorial Class!

- User Guide
  - cd $TRICK_HOME/docs
  - firefox index.html

- The Source and Include files can be found on the CD
  - root/trick_models/copies/gravity/include
  - root/trick_models/copies/gravity/src
- The simulation files can be found on the CD
  - root/trick_sims/SIM_cannon_copy/*

- Trick Website
  - http://trick.jsc.nasa.gov/