Expanding Utility of RECAST-wf For Particle Simulations

Ed van Bruggen
Prof. Shih-Chieh Hsu
Alex Schuy, Lukas Heinrich, Vladimir Ovechkin, Colton Peterson

IRIS-HEP 2021 Fellowship
What are Particle Simulations?

- Particle physics experiments at LHC are extremely expensive.
- Simulate collisions of real particle collisions at LHC and ATLAS detector to test new possible theories.
- Full particle simulations would also simulate how the final products interact with detectors:
  - Truth-level simulations ignore this to reduce complexity.
  - Sacrifices accuracy and completeness for speed and ease of use.
RECAST

- Instead of starting from beginning for each new model, repurpose old analyses
- **RECAST** allows previously used complex analyses to be reinterpreted to new physics models and simulations
- The Standard Model calculations of an analysis are the same for most uses
  - Just need to estimate the effects of the new physics
- Modularize and preserves new analyses for future uses after original author
RECAST-workflow

- **Python package and command line tool** for creating fast workflows to test and reproduce truth level simulations
- Creates workflows to be carried out by yadage
  - Manages series of subworkflows, handling passing of input variables/files and output
- Uses Docker to run each program in its own container preinstalled and correctly configured
- Eventual goal to run on Reana supercomputer cluster via web interface
  - Nothing needs to be installed or configured locally, use web interface to upload config files
- Building on previous work of Vladimir Ovechkin and Alex Schuy
MadGraph

- Simulation tool for doing event generation of initial particle collision
- Allows inclusions of model of new physics models (UFO)
- Define Feynman diagrams and model parameters
- We improved MadGraph integration in RECAST-wf to support custom physics models and parameters
Pythia and Herwig

- Two different simulation programs which perform showering simulations from MadGraph events
- Combines parton level events from MadGraph to hadron output
- Pythia only does the showering for MadGraph events
  - Can be run directly from MadGraph
- **Herwig** is an alternative event generator which also includes showering
  - Does not support BSM models like pythia, need to import model
- Pythia previously integrated MadGraph subworkflow, now modularized into new showering step so can be swapped in Herwig
- Added Herwig showering support to RECAST-wf
Pythia and Herwig Workflows in RECAST: Created new showering step
MadGraph Custom Parameters

steer_madgraph.py:

def edit_proc_card(proc_card_path, param_card_path, run_card_path, shower_card_path, shower, n_events):
    with open(proc_card_path, 'a') as f:
        f.write('output output
')
        f.write('launch
')
        if shower:
            f.write('shower=Pythia8\ndone
')
        else:
            f.write('shower=OFF\ndone
')
        # Read from the given cards and set the number of events.
        if param_card_path is not None and param_card_path != "default":
            with open(param_card_path, 'r') as card:
                f.write(card.read())
                f.write('
')
        if n_events is not None:
            f.write('set events {}
'.format(n_events))
        f.write('0
')
steer_madgraph.py:

```python
if ufotar is not None and ufotar != "default":
    if '/' in ufotar:
        ufo_name, ufo_ext = os.path.splitext(os.path.basename(ufotar))
    else:
        ufo_name, ufo_ext = os.path.splitext(ufotar)
    assert ufo_ext == '.tar', 'ufo must be a tar file!'
    ufo_name = ufo_name.strip('.ufo')
    subprocess.call(['tar', '-xf', ufotar])
    ufo_path = '/code/madgraph/models/{}'.format(ufo_name)
    shutil.rmtree(ufo_path, ignore_errors=True)
    shutil.copytree(ufo_name, ufo_path)
    proc_card_copy_path = os.path.join(os.getcwd(), 'proc_card.dat')
    shutil.copyfile(proc_card_path, proc_card_copy_path)
    edit_proc_card(proc_card_copy_path, param_card_path,
        run_card_path, shower_card_path, shower, n_events)
    subprocess.call(['/code/madgraph/bin/mg5_aMC', proc_card_copy_path])
    run_dir = os.path.join(os.getcwd(), 'output', 'Events', 'run_01')
    event_path = glob.glob(os.path.join(run_dir, '*._lhe.gz'))[0]
    subprocess.call(['gunzip', event_path])
    event_path = glob.glob(os.path.join(run_dir, '*._lhe'))[0]
    subprocess.call('cp {} {}'.format(event_path, output_path), shell=True)
```
Separate Pythia Into New Showering Step

description.yml:

```yaml
description: 'Perform parton showering through Pythia 8.'
environment_settings:
  - {name: pythia_version, default: '8240'}
build_tags:
  - pythia: pythia_version
inputs:
  - {name: n_events, description: 'The number of events that should be generated.'}
  - {name: pythia_card, description: 'Pythia settings.', optional: true}
interfaces:
  input:
    - lhe
  output:
    - hepnc
```

pythia.yml:

```yaml
process:
  process_type: string-interpolated-cmd
  cmd: 'python steer_pythia.py {lhe} {outputhepnc} {n_events} {pythia_card}'
publisher:
  publisher_type: "frompar-pub"
  outputmap:
    hepnc: outputhepnc
environment:
  environment_type: "docker-encapsulated"
  image: recast/pythia
  tmagetag: {pythia_version}
```
Support Custom Pythia Card

steer_pythia.py:

def runpythia(inputlhe, outputlhepmc, nevents, pythiacard):
    if pythiacard is None or pythiacard == "default" or pythiacard == "null":
        pythiacard = 'pythia.tmpl'
        runcardname = 'pythia_card.dat'
    with open(pythiacard, 'r') as template:
        with open(runcardname, 'w+') as filled:
            filled.write(template.read().format(
                LHEF=inputlhe, NEVENTS=nevents))
    subprocess.check_call(shlex.split(
        "./pythia8/examples/main42 {} {}\".format(runcardname, outputlhepmc)))
New Herwig Subworkflow

description.yml:

```yaml
description: 'Perform parton level showering through Herwig.'
environment_settings:
  - {name: herwig_version, default: '7.2.1'}
build_tags:
  herwig: herwig_version
inputs:
  - {name: n_events, description: 'The number of events that should be generated.'}
  - {name: herwig_file, description: 'The IN card for Herwig.'}
  - {name: herwig_model, description: 'The model file for BSM Herwig.'}
  - {name: herwig_so, description: 'The shared object file for the Herwig model.'}
interfaces:
  input:
    - lhe
  output:
    - hepmc
```
Herwig Subworkflow

herwig.yml:

```yaml
process:
  process_type: interpolated-script-cmd
  script: |
    cp {herwig_model} ./FRModel.model-
    cp {herwig_so} ./FRModel.so-
    cp {lhe} ./events.lhe-
    export LD_LIBRARY_PATH=/herwig/lib:/herwig/lib64-
    bash herwig-run.sh {herwig_file} {n_events} -
    cp output.hepmc {outputhepmc} -
publisher:
  publisher_type: "frompar-pub"
outputmap:
  hepmc: outputhepmc-
environment:
  environment_type: "docker-encapsulated"
image: recast/herwig-
imagetag: {herwig_version} -
```

herwig-run.sh:

```bash
#!/bin/bash-
-
INFILE="$1" -
RUNFILE=$(basename "$1".run) -
EVENTS=${2:-10000} -
-
/herwig/bin/Herwig read "$INFILE" -
/herwig/bin/Herwig run "$RUNFILE" -N $EVENTS -
```
Mono-sbb Model Interpretation

- New Beyond Standard Model proposal which includes Dark Matter
- Dark Matter is theorized new particle which explains astronomical observations
- Mono-sbb would allow for Dark Matter detection at LHC proton-proton collisions
  - Reinterpretation of previous Higgs Boson model
- Requires new force mediator (Z’) and Dark Higgs Boson (s)
  - Dark Higgs decays into two bottom quarks which can be measured by ATLAS detector
  - If b-jets are detected with less energy than expected it would provide indirect evidence for Dark Matter
Results

- Plots of transverse momentum and missing transverse energy that we would expect to see if model is true
Results

Contour plots of parameter scan showing where model is explained by the Standard Model

- \( \sqrt{s} = 13 \text{ TeV}, \ 36.1 \text{ fb}^{-1} \)
- \( g_q = 0.25, \ g_x = 1.0, \)
- \( m_x = 200 \text{ Gev} \)
Summary

- Extended MadGraph support for custom models
- Separated Pythia into new showering step, support custom parameters
- Added Herwig showering
- Applied Pythia and Herwig showering to mono-Sbb model
- Implemented Sherpa showering
- Expanded documentation

Future work:
- Complete Sherpa to study a new model
- Add generator support for Herwig and Sherpa
- Run on Reana through web interface
References

6. Herwig: [https://herwig.hepforge.org/](https://herwig.hepforge.org/)

Special thanks to: Shih-Chieh Hsu, Alex Schuy, Lukas Heinrich, Vladimir Ovechkin, Colton Peterson
New Sherpa Subworkflow

description.yml:

```
description: 'Perform parton level showering through Sherpa.'
environment_settings:
  - {name: sherpa_version, default: 'latest'}
built_tags:
  sherpa: sherpa_version
inputs:
  - {name: n_events, description: 'The number of events that should be generated.'}
  - {name: sherpa_file, description: 'The input card for Sherpa.'}
interfaces:
  input:
    - lhe
  output:
    - hepmc
```

sherpa.yml:

```
process:
  process_type: interpolated-script-cmd
  script:
    - cp {lhe} ./events.lhe
    - bash sherpa-run.sh {sherpa_file} {n_events} 
    - cp output.hepmc {outputhepmc}
publisher:
  publisher_type: "frompar-pub"
  outputmap:
    hepmc: outputhepmc
environment:
  environment_type: "docker-encapsulated"
  image: recast/sherpa
  imagetag: {sherpa_version}
```