



Science and
Technology
Facilities Council

Welcome

Image © Haroon Rafique





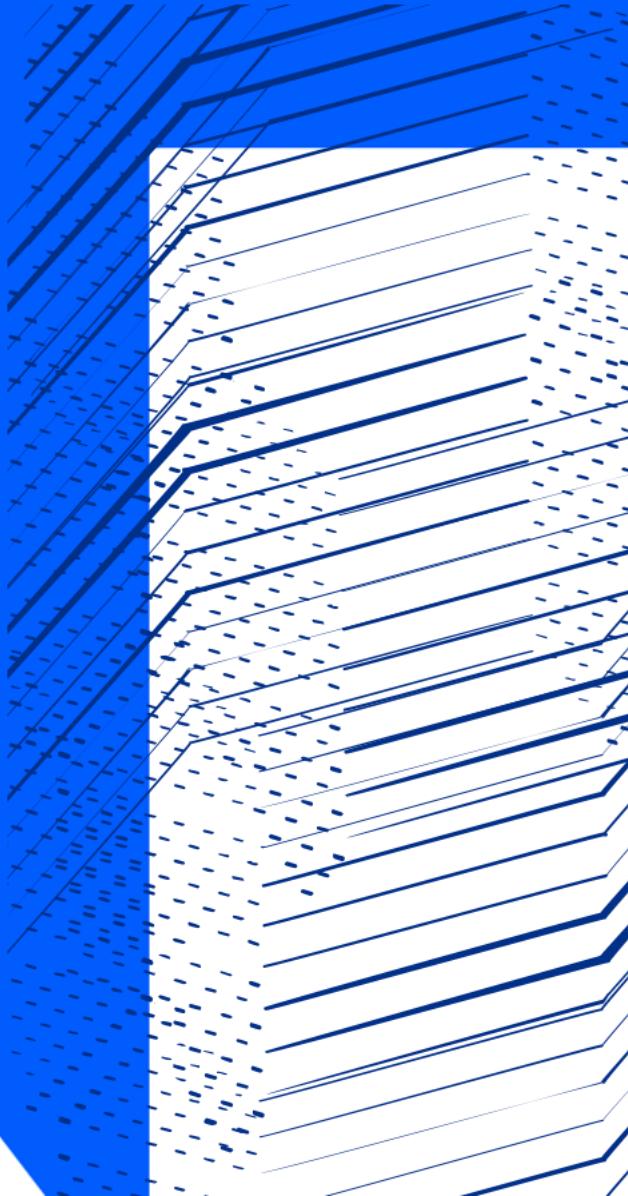
Science and
Technology
Facilities Council

Accelerator Simulation Framework

Towards end-to-end beam dynamics
simulations of the ISIS accelerators

Haroon Rafique

haroon.rafique@stfc.ac.uk



Today's Journey

0 Introduction

Who's the new guy and what can he help me with?

1 Common Issues

Saving time. Using each others work. **Doing physics rather than programming.**

2 Approaches

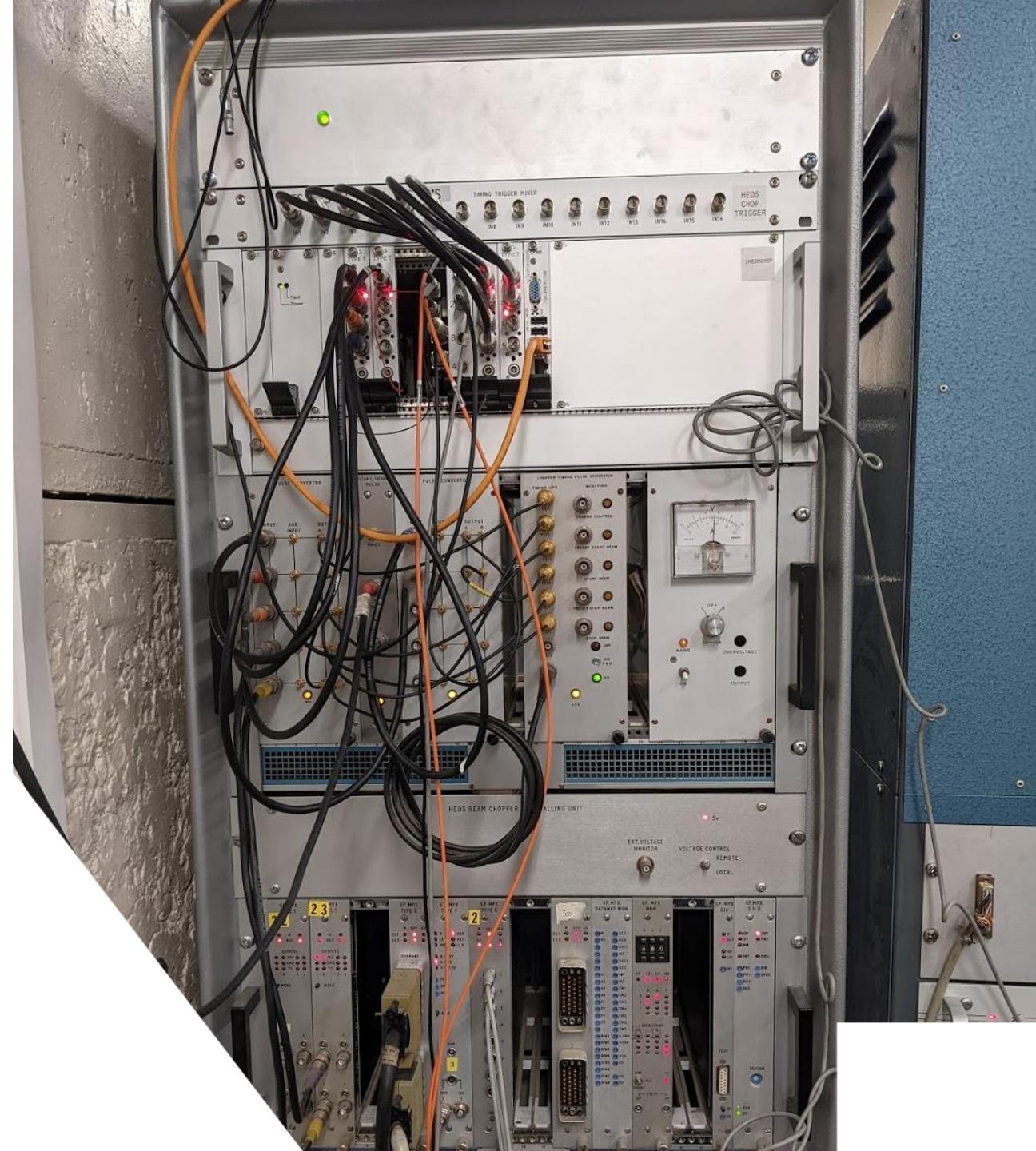
Lessons learnt from personal experience.

3 Opportunities

A vision of the future based on the efforts of the past.

4 Working Together

If we work from the same foundation, we can build a future-proofed, dynamic, and powerful model. Time for debate.



Haroon Rafique

2007-11: MPhys @ Manchester

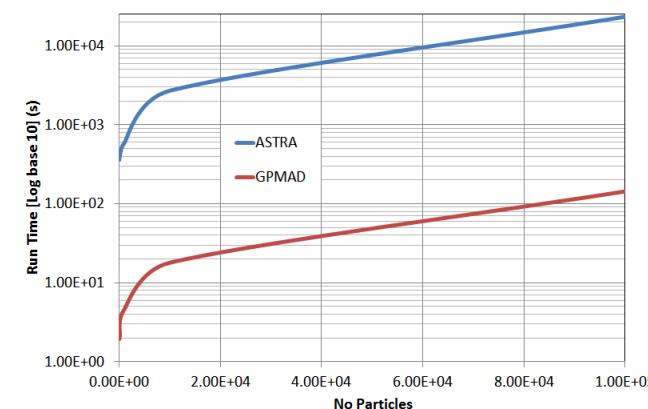
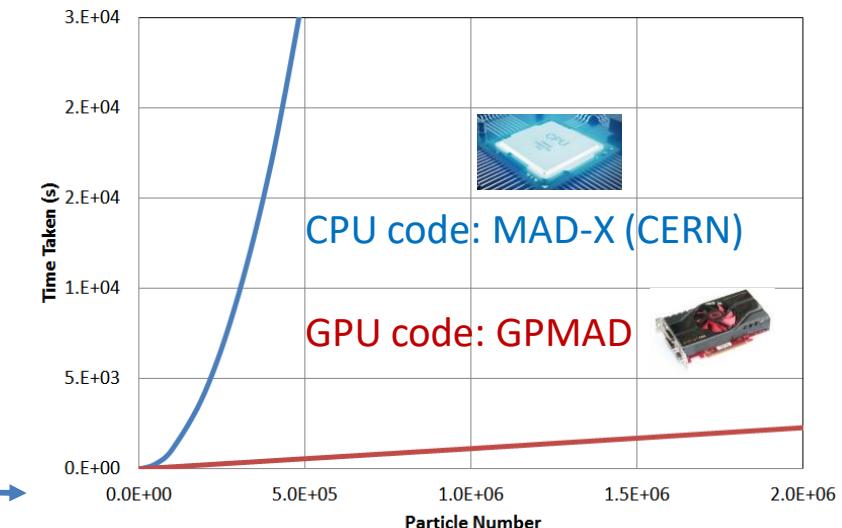
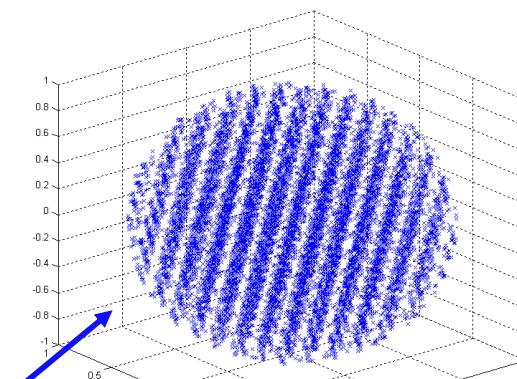
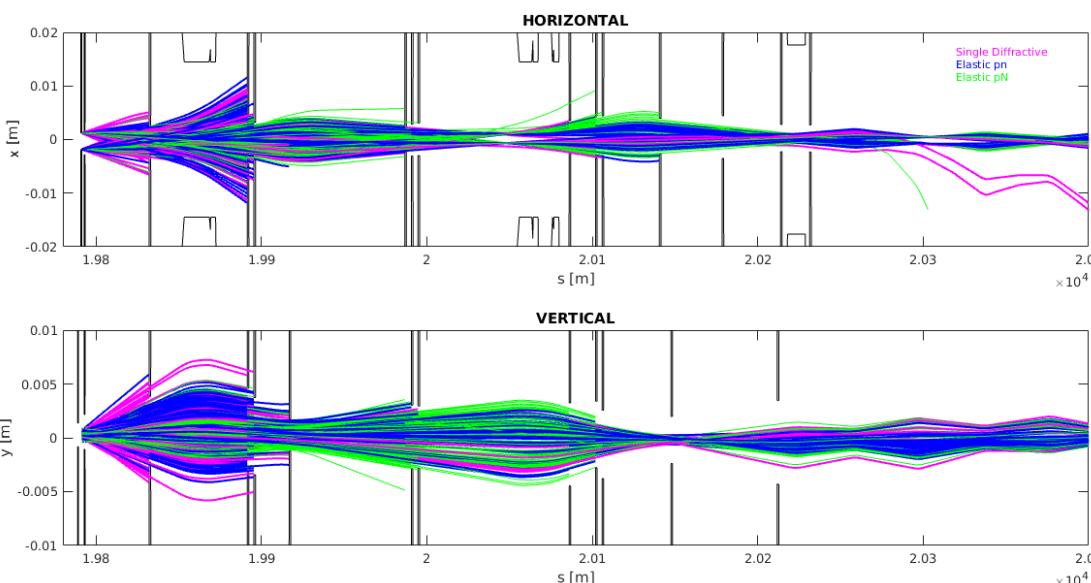
Prof. Rob Appleby, Dr. Hywel Owen

2nd Year theory project: random numbers aren't always random

MPhys project: GPUs make particle tracking very fast

Even with space charge (frozen simple)

One of the first scientific uses of CUDA!



2011-16: PhD @ Huddersfield / NGACDT

Prof. Roger Barlow [in collaboration with CERN]

MERLIN for HL-LHC Collimation

Loss maps, novel composite material scattering, hollow electron lens collimation

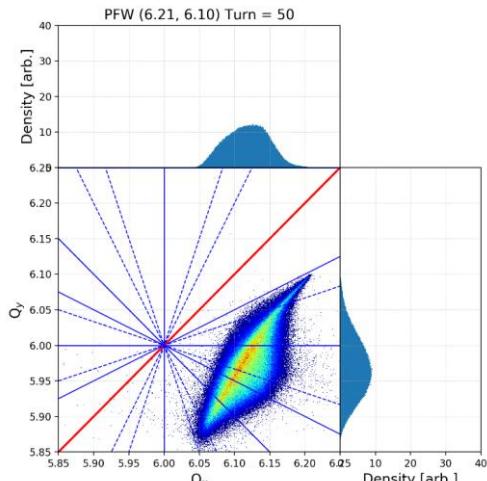
Thesis published in 2017 because it was too long – keep it short and sweet to save time

Haroon Rafique

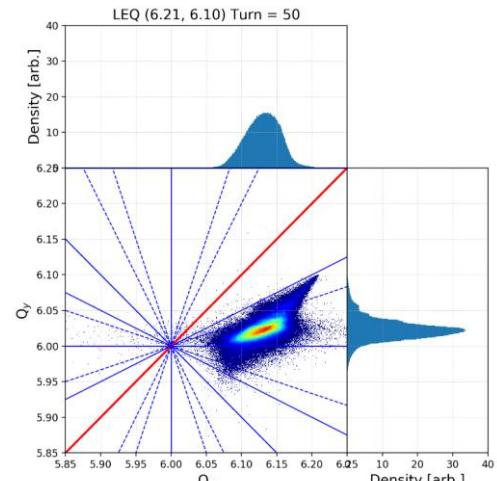
2016-18 PDRA @ Manchester

Prof. Rob Appleby [in collaboration with CERN/JAI/EuroCirCoL]

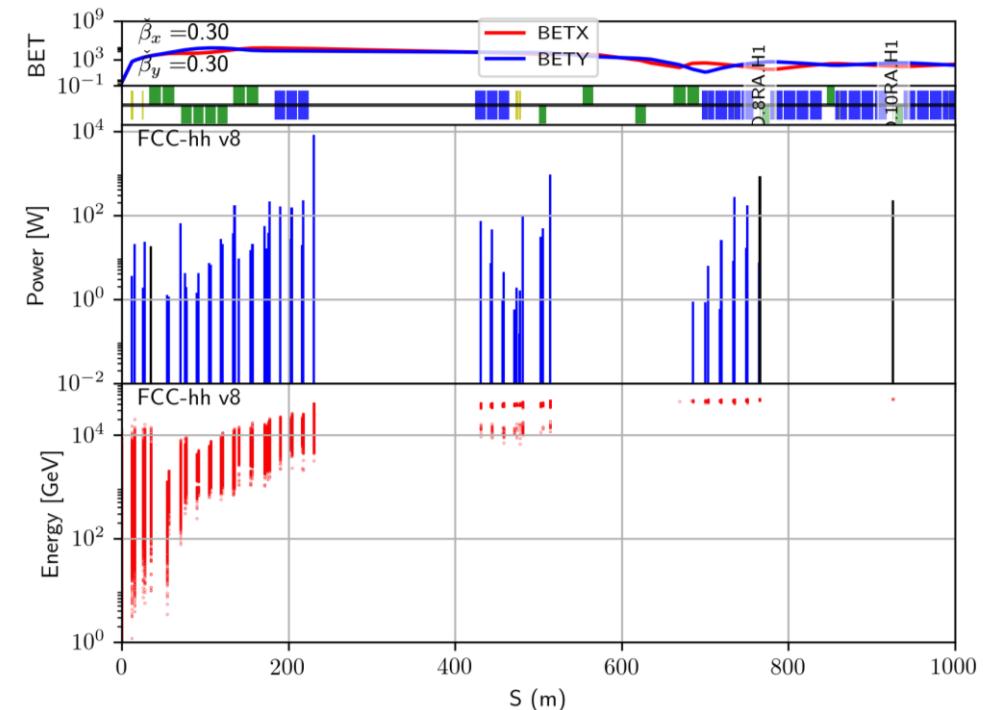
FCC-hh 50 TeV collision debris: reduced damage to SC magnets by adding collimators to design (specifications checked in FLUKA)
Muon tracking (analytic & FLUKA) shows no 20 TeV muons reach next detector through standard rock or accelerator tunnel



Control working point with main magnet auxiliary windings



Control working point with low energy quads



2018-20 Senior Fellow @ CERN

Dr. Hannes Bartosik, Dr. Yannis Papaphilipou, Dr. Alex Huschauer

Space Charge @ Proton Synchrotron Injection:

Large horizontal emittance growth between Booster & PS (30-40%)

Implemented infrastructure to perform PTC PyORBIT PIC simulations including:

- Space charge benchmarking campaign with profile measurements
- Measured injection bump tune swing caused by eddy currents
- Measured dispersion mismatch in transfer line
- Initial distribution generated using measurements
- Installation and support on HPC infrastructure (2nd largest HPC user at CERN at >1180 CPU years)

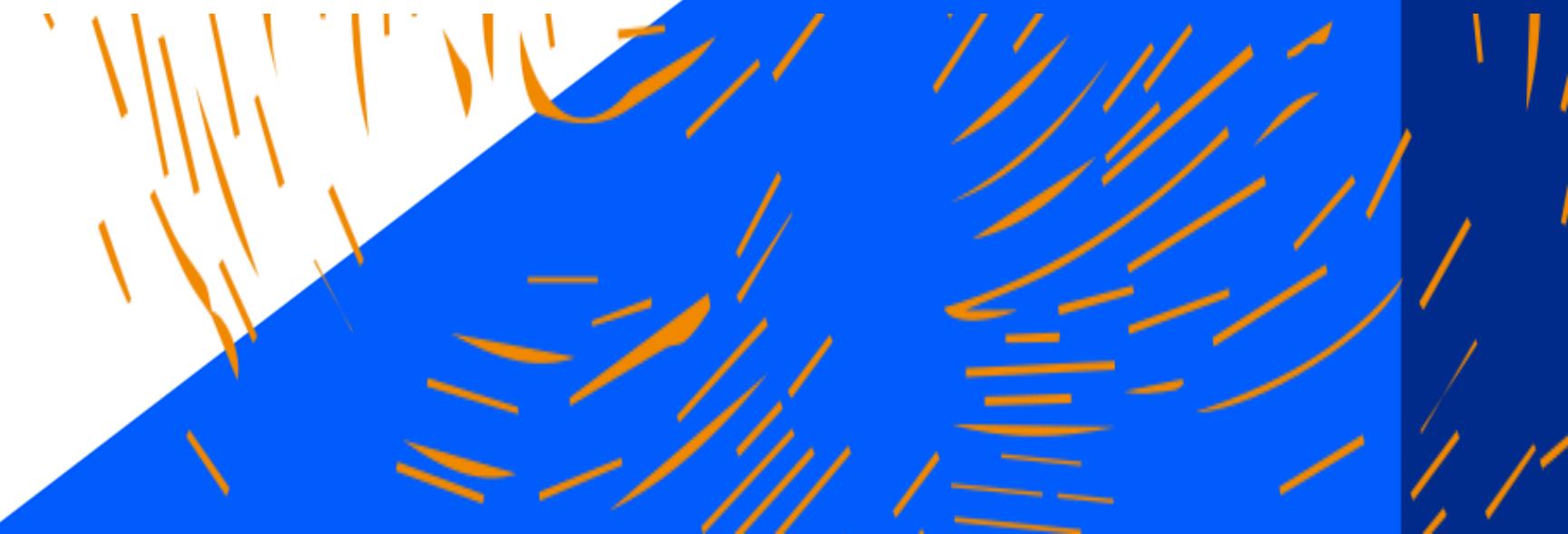
Used to show that **space charge is not expected to be a major contributor to the 30-40% emittance growth**, and possible benefits of alternative tune control in operation. Used to predict performance for upgrades (now in commissioning).



Science and
Technology
Facilities Council

Common Issues

Common to me at least



What do we need when starting a new accelerator project?

Beam

- **Parameters:** Energy, energy spread, charge, intensity etc
- **What does it look like:** 6D Profiles/distributions, measurements etc...

Accelerator Lattice (for operational machine)

- **Sequence of lattice elements:** what is in the accelerator and where
- **Magnet strengths:** constants, ramp functions – what changes and when

Specifics

- **For example:** apertures, operational settings, inputs/outputs for other codes...

Where do we get this information?

People

- **Point me:** provide references
- **Explain to me:** based on experience
 - **Requires: People and their time**

References

- **Documentation:** what year is it from? Is it still relevant? **Costs time to read and search for information**
- **Examples:** code snippets, simulations etc. **Usually require explanation from a person or some time to understand (familiar language etc?)**

Blind Search

- **Have a rummage:** look through the shared network drive
 - **Requires: time and access**

Modelling Complex Facilities

It makes sense to divide modelling into expert areas:

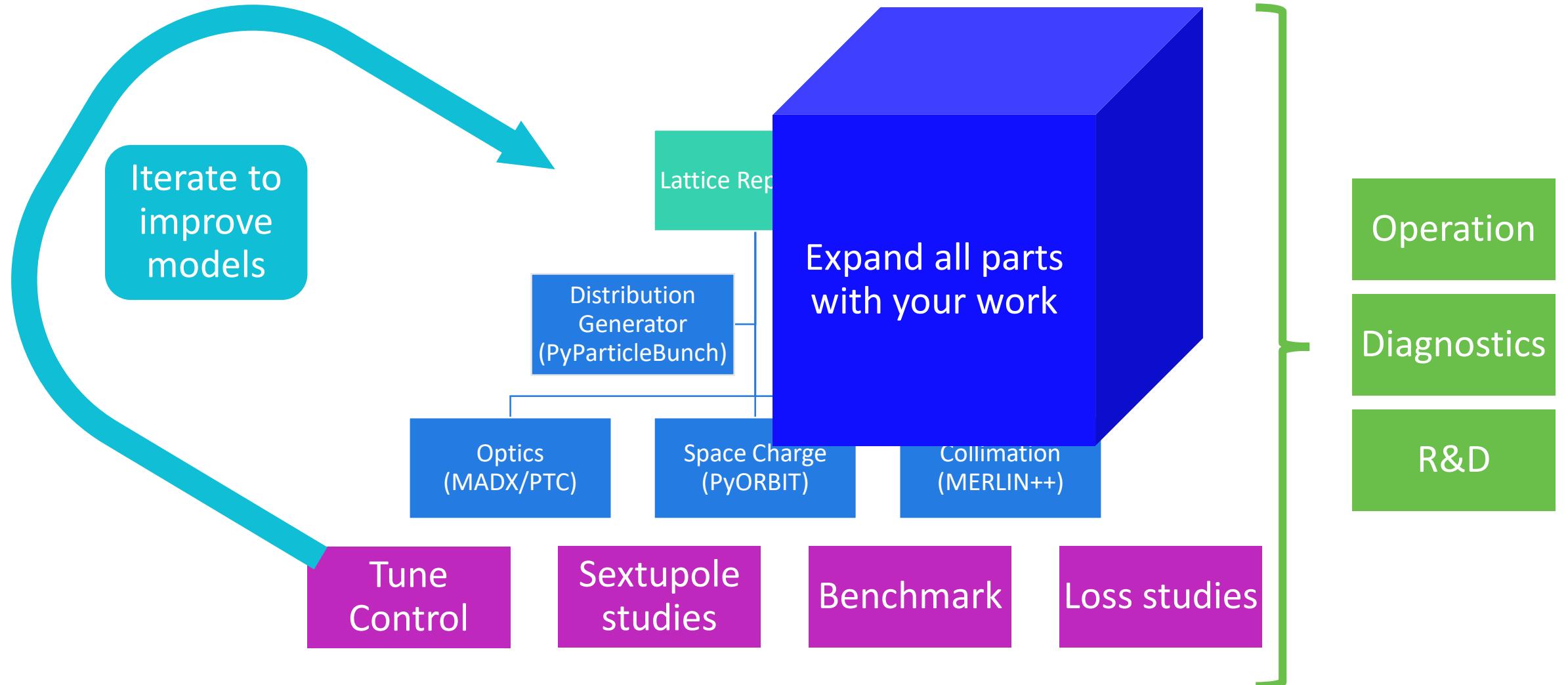
- Ion Source
 - RFQ
 - Linac
 - Transfer lines
 - Synchrotron
 - Vacuum systems
 - Power & RF systems
 - Specific systems (e.g. charge exchange injection)

How do we bring the relevant parts together easily?



Why? We're not that far from an end-to-end ISIS accelerator simulation...

Proposal: Use a shared simulation framework



How do we work together?

Individual models/methods/codes/scripts

- How and where should I **store** important data/information/code/scripts?
- How can I develop my own scripts/code in a way that can be **accessible** to others, avoids **problems with unfinished work, bugs** etc, and **isn't difficult?**

Group framework

- How should we **structure** everything? Should it be different for **design/operation?**
- What **codes** should we use and **why?** Do we have **alternative codes?**
- How do we all use the same thing but modify our own versions **without headache?**

Group consensus

- Everyone has their **own valid methods**
- The idea is to **share and standardise to make our lives easier** not re-do everything

Benefits of a shared simulation framework



- **Consistency:** everyone is using the same model
- **Reproducibility:** important for collaboration
- **Reduced input, improved output**
- **Save time:** from new starters to big simulations
- **Bug protection:** revision control has many benefits
- **Fool-proof:** Provide **standard methods** for bridging codes/models

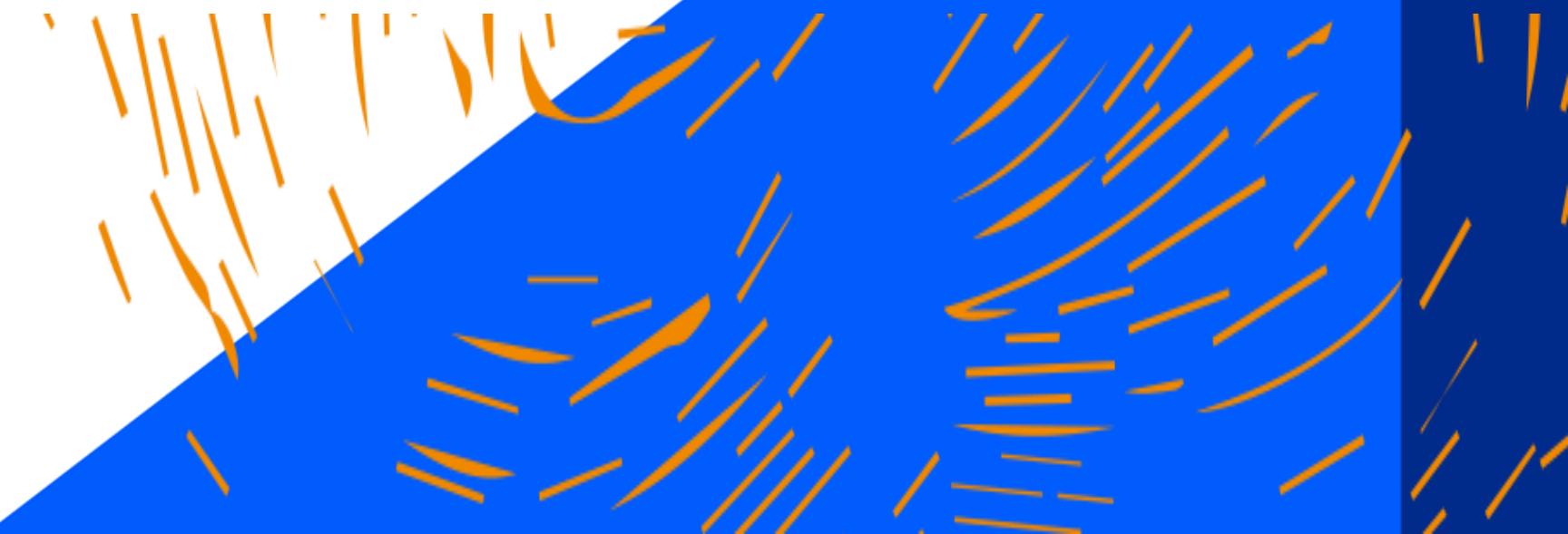
Also pandemic proof!



Science and
Technology
Facilities Council

Approaches

How do we resolve common issues?



Start with a strong foundation

Git is software for **tracking changes in any set of files**, usually used for **coordinating work among programmers collaboratively developing source code** during software development.

Its goals include **speed, data integrity, and support for distributed, non-linear workflows**.

GitHub: Microsoft owned public service

GitLab: Open core company offered service (has more advanced features)

HR's GitHub: <https://github.com/HaroonRafique>

HR's STFC GitLab: <https://gitlab.stfc.ac.uk/mmf67433>

See the following talk for more details on using Git



STFC GitLab



Science & Technology
Facilities Council

Open source software to collaborate on code

Manage Git repositories with fine-grained access controls that keep your code secure. Perform code reviews and enhance collaboration with merge requests. Each project can also have an issue tracker and a wiki.

You can view our terms and conditions [here](#).

STFC has its own (secure) GitLab

STFC Federal ID	Standard
STFC Federal ID Username	
<input type="text"/>	
Password	
<input type="password"/>	
<input type="checkbox"/> Remember me	
<input type="button" value="Sign in"/>	

ISIS Accelerator Lattice Repository

https://gitlab.stfc.ac.uk/isis_accelerator_models (full tree)

https://gitlab.stfc.ac.uk/isis_accelerator_models/isis

The screenshot shows the GitLab interface for the 'isis_accelerator_models' group. The top navigation bar includes links for JK SBS, CodiMD - Collabor..., ISIS-II, Teams, Sign In, ISIS Intranet, Groups, More, and a search bar. Below the navigation is a sidebar for 'isis_accelerator_models' with sections for Subgroups and projects, Shared projects, Archived projects, and a search bar. The main content area lists four projects under 'Subgroups and projects': EPB, EPB2, HEDS, and Isis. Each project has a thumbnail icon, a name, a star rating (0), and a timestamp (4 weeks ago). A 'New subgroup' and 'New project' button is located at the top right of the project list.

The screenshot shows the GitLab repository page for the 'isis' project. The top navigation bar includes links for JK SBS, CodiMD - Collabor..., ISIS-II, Teams, Sign In, ISIS Intranet, Groups, More, and a search bar. The repository details are shown: Project ID: 2222, 16 Commits, 1 Branch, 0 Tags, 12.3 MB Files, 12.3 MB Storage. The description is 'Accelerator lattice models for the existing ISIS synchrotron'. The repository interface includes tabs for master, isis, History, Find file, Web IDE, and Clone. A commit history table shows the following entries:

Name	Last commit	Last update
00_Lattice_Files	<add> [00_Lattice_Files] initial lat3ud files m...	2 days ago
00_Scripts	<update> [00_Scripts]	1 day ago
01_Injection	<README> [01:madx]	1 day ago
.gitignore	<git> [all] README and .gitignore added	2 days ago
README.md	<update> [README]	2 days ago
madx-linux64-v5_06_01	<added> [madx-linux64-v5_06_01]	1 day ago

A 'README.md' section is also visible with the heading 'ISIS Synchrotron Lattice Repository' and a note about the database for MAD-X based ISIS Synchrotron lattice including example scripts in MAD-X and cpymad format. A bulleted list of files and their descriptions follows:

- **00_Scripts:** Global helper scripts
- **00_Lattice_Files:** MAD-X input scripts
- **01_Injection:** 70 MeV injection (bare) lattice
- **02_Full_Cycle:** Implementation of magnet cycle in arbitrary steps from 70 MeV - 800 MeV
- **03_Closed_Orbit:** Examples of closed orbit excursions, corrections, bumps etc
- **04_Tune_Control:** Implementation of realistic tune control
- **05_Errors:** Implementation of single/distributed strength/alignment errors
- **06_Magnet_Splitting:** Implementation of magnet splitting
- **07_Envelopes:** Matching the lattice to measurements

ISIS Accelerator Lattice Repository: How to use

0 Request access on the STFC GitLab

Use secure internal GitLab for work, in future we can copy things to Github for ease of access/teaching etc

1 Clone the repository onto your machine

Create a local copy for you to use

2 Run the examples to check you have the requirements

Python, ipython notebook, cymad, tfs, madx, etc

3a Basic uses (if you add/upgrade features consider sharing):

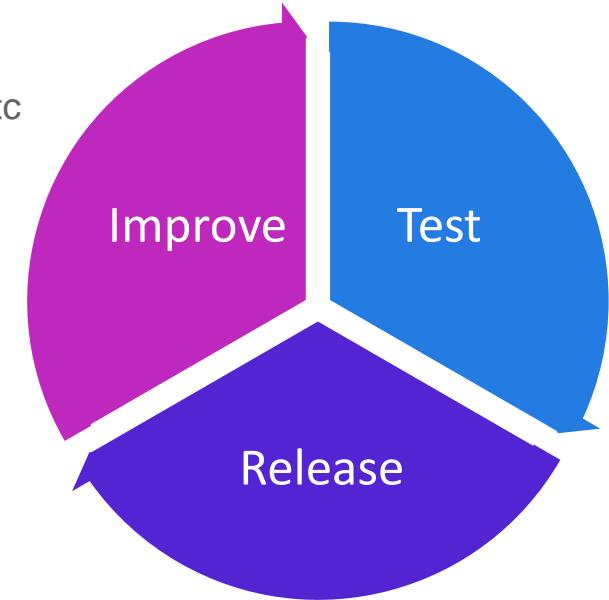
- Generate MAD-X/PTC format inputs for another code (e.g. PyORBIT)
- Look at optics changes; closed orbit errors, gradient errors etc...

3b Advanced uses (fork/branch the code to create your own copy first):

- Implement your fancy scripts, where possible breaking them down into universal scripts we can all use – ONLY work that is relevant to the lattice repository (MAD – related lattice modifications).
- Example: Pete's tune control mapping, Hayley's optics measurements, ... (I'll help of course)

4 Create a pull request (pull your work into the main branch for everyone)

Short and sweet: Your modifications, why they are needed, your proof that they don't break everything and work as expected



ISIS Accelerator Lattice Repository: cpymad example

Folder:

https://gitlab.stfc.ac.uk/isis_accelerator_models/isis/-/blob/master/01_Injection/cpymad/

iPython Notebook:

https://gitlab.stfc.ac.uk/isis_accelerator_models/isis/-/blob/master/01_Injection/cpymad/cpymad_ISIS_Injection.ipynb

Useful link: Jupyter Online Notebook Viewer
(note this won't work with STFC GitLab
notebooks as they aren't public):

<https://nbviewer.jupyter.org/>

Time for an interactive example...

The screenshot shows a Jupyter Notebook interface on a GitLab page. The left sidebar lists project sections like Project overview, Repository, Files, Commits, Branches, Tags, Contributors, Graph, Compare, Issues, Merge Requests, CI / CD, Operations, Packages & Registries, Analytics, Wiki, Snippets, Members, and Settings. The main area displays a notebook titled 'cpymad_ISIS_Injection.ipynb'. The code in the notebook includes imports for os, numpy, and pandas, a section on importing MAD-X from CPYMADE, and examples of using helper functions and standard matplotlib plot parameters.

```
import os
import numpy as np
import pandas as pd

from cpymad.madx import Madx
import tfs

%run '../../../../../00_Scripts/helper_functions.py'

plt.rcParams['figure.figsize'] = [8.0, 5.0]
plt.rcParams['figure.dpi'] = 200
plt.rcParams['savefig.dpi'] = 200

plt.rcParams['axes.titlesize'] = 14
plt.rcParams['axes.labelsize'] = 14
```

ISIS Accelerator Lattice Repository: Ideas

Optics:

Static:

- Generate matched lattice for given parameters/constraints
- Investigate closed orbit errors/correction
- Investigate gradient errors/correction
- Benchmark lattice models using measurements

Dynamic:

- Generate magnet ramping tables
- Generate matched lattice for given parameters/constraints over time

Operation:

- Improved and up to date model for apps such as LOCO
- Starting point for implementing corrections such as Pete's advanced tune controls

Simulation:

- Standard methods to produce MADX/PTC etc outputs required as input to other codes
- Use together with 'accelerator beam distribution' repository to perform bunch tracking in MAD-X and PTC

Lattice Repository: How to start developing

1. Sign in to create an STFC GitLab account: <https://gitlab.stfc.ac.uk/>
2. Request access to the Project: https://gitlab.stfc.ac.uk/isis_accelerator_models
3. Get stuck in using the issues boards:
 - General lattice repo stuff:
https://gitlab.stfc.ac.uk/groups/isis_accelerator_models/-/boards
 - Specific ISIS synchrotron repo stuff:
https://gitlab.stfc.ac.uk/isis_accelerator_models/isis/-/boards/80
4. Assign tasks to yourself and start forking/branching/pushing/pulling etc!

Lattice Repository: Group Page

We start at Details

The screenshot shows a GitLab group page for 'ISIS_Accelerator_Models' (Group ID: 781). The left sidebar lists group details, activity, issues (1), merge requests (0), Kubernetes, packages & registries, members, and settings. The main area displays four projects: EPB, EPB2, HEDS, and Isis. The Isis project is highlighted with a pink border. A green arrow points from the 'Details' link in the sidebar to the Isis project in the list.

Project	Description	Rating	Last Updated
EPB	Accelerator lattice models for the extracted proton beamline (EPB) to target station 1	★ 0	1 month ago
EPB2	Accelerator lattice models for the second extracted proton beamline (EPB2) to target ...	★ 0	1 month ago
HEDS	Accelerator lattice models for the high energy drift space (HEDS) between linac tank 4...	★ 0	1 month ago
Isis	Accelerator lattice models for the existing ISIS synchrotron	★ 0	1 month ago

Let's take a look at the ISIS synchrotron lattice repository

Each lattice has it's own repository

Lattice Repository: ISIS Synchrotron Page

The screenshot shows a GitLab project page for 'Isis'. The left sidebar has a 'Details' section highlighted with a blue arrow and the text 'We start at Details'. Below it, the 'Issues' section is highlighted with a pink box and the text 'Let's take a look at the issue board'. A green arrow points from the 'File contents of the repository' text to the 'Repository' section in the sidebar. Another green arrow points from the 'Markdown README under this' text to the 'README' button in the top right of the main content area.

We start at Details

Let's take a look at the issue board

File contents of the repository

Markdown README under this

ISIS Accelerator Models > Isis

Isis Project ID: 2222

22 Commits 1 Branch 0 Tags 22.2 MB Files 22.2 MB Storage

Accelerator lattice models for the existing ISIS synchrotron

master isis / + History Find file Web IDE Clone

<update>[02: Beta function matching]
Haroon Rafique authored 4 days ago
333d8050

README Add LICENSE Add CHANGELOG Add CONTRIBUTING Enable Auto DevOps Add Kubernetes cluster
Set up CI/CD

Name	Last commit	Last update
00_Lattice_Files	<update>[02] added to cpymad notebook	5 days ago
00_Scripts	<update>[00: helper functions]	6 days ago
01_Injection	<update>[00, 01]	1 week ago
02_Beta_Function_Matching/cpy...	<update>[02: Beta function matching]	4 days ago
.gitignore	<git>[all] README and .gitignore added	1 week ago
README.md	<update>[README]	1 week ago
madx-linux64-v5_06_01	<added>[madx-linux64-v5_06_01]	1 week ago

Lattice Repository: ISIS Synchrotron Issue Board

The screenshot shows a GitLab issue board titled "Development" for the project "ISIS Accelerator Models > Isis". The board has five columns:

- suggestion**: Contains one card labeled "Have an idea?".
- idea**: Contains one card labeled "New feature/upgrade".
- enhancement**: Contains one card labeled "Fix master lattice".
- documentation**: Contains one card labeled "Documentation request/idea/requirement".
- example**: Contains one card labeled "Example request/idea/requirement".

Each card has a green diagonal arrow pointing towards it from the bottom right. Below the board, there are instructions:

- Write 'cards' in markdown
- Drag and drop
- Assign tasks to members
- Assign additional labels – help-wanted

Lattice Repository: Labels

Labels can be applied to issues and merge requests.

Star a label to make it a priority label. Order the prioritized labels to change their relative priority, by dragging.

Prioritized Labels

idea

A thought or suggestion as to a possible course of action

Issues · Merge requests ·

Prioritized label



Unsubscribe

enhancement

A new feature or improvement to an existing feature

Issues · Merge requests ·

Prioritized label



Unsubscribe

bug

Something is not behaving as expected

Issues · Merge requests ·

Prioritized label



Unsubscribe

example

Please provide an example of how to use this

Issues · Merge requests ·

Prioritized label



Unsubscribe

documentation

Please can you explain how this works

Issues · Merge requests ·

Prioritized label



Unsubscribe

ISIS_Accelerator_Models

Lattice Repository: Labels

Other Labels

discussion

This is up for debate

Issues · Merge requests



Unsubscribe

help-wanted

I need help with something

Issues · Merge requests



Unsubscribe

Level: critical

Needs immediate attention: main features are affected

Issues · Merge requests



Unsubscribe

Level: low

Problem does not affect code features: e.g. incomplete documentation

Issues · Merge requests



Unsubscribe

Level: moderate

Problem only affects non-critical features or extended functions: e.g. change in external code requires update for extra features of external code

Issues · Merge requests



Unsubscribe

suggestion

Maybe this would work better a different way. Maybe not

Issues · Merge requests



Unsubscribe

github.com/users/HaroonRafique/projects/1

Search or jump to... Pull requests Issues Marketplace Explore

Accelerator Simulation Framework Project

Accelerator_Simulation_Framework
Updated 5 minutes ago

Idea

- PyParticleBunch::Formats::PTC
Added by HaroonRafique
- PyParticleBunch::Formats::MADX
Added by HaroonRafique
- PyParticleBunch::Distributions::Poincare
Added by HaroonRafique
- PyParticleBunch::Distributions::Uniform
Added by HaroonRafique
- PyParticleBunch::Distributions::Joho
Added by HaroonRafique
- PyParticleBunch::Distributions::Waterbag
Added by HaroonRafique
- PyParticleBunch::Distributions::KV
Added by HaroonRafique
- PyParticleBunch::Distributions::Gaussian
Added by HaroonRafique

To do

- Examples:
0 of 4
ParticleBunch#3 opened by HaroonRafique
bug enhancement help wanted
- Generate a particle distribution
0 of 9
ParticleBunch#1 opened by HaroonRafique
bug enhancement help wanted

In progress

- Document physics processes behind particle generation and normalisation used in the code:
0 of 3
ParticleBunch#2 opened by HaroonRafique
bug enhancement help wanted
- Create Git usage instructions
Accelerator_Simulation_Framework#2 opened by HaroonRafique
documentation good first issue help wanted
- Create a Git style guide
0 of 1
Accelerator_Simulation_Framework#1 opened by HaroonRafique
documentation good first issue help wanted

Done

- + Add column

Filter cards

HaroonRafique commented 2 minutes ago

Browser

- Edit markdown files

Linux

- Basic Git commands
- Advanced Igit commands

Windows

- Git GUI instructions ?

Show less

Assignees

Labels

- documentation good first issue help wanted

Projects

- Accelerator_Simulation_Framework In progress

Milestone

No milestone

Go to issue for full details

Close issue

Markdown let's us make checklists

GitHub (public) because:

- PyORBIT/MERLIN etc on GitHub
- Doesn't contain sensitive code
- Easier to access

Let's take a look at the full issue page

Accelerator Simulation Framework: Git usage instructions issue

The screenshot shows a GitHub issue page for a repository named 'HaroonRafique / Accelerator_Simulation_Framework'. The issue is titled 'Create Git usage instructions #2' and is currently open. The page includes a detailed description of the required steps for different operating systems (Browser, Linux, Windows) and a note about Git GUI instructions. The issue has been labeled with 'documentation', 'good first issue', and 'help wanted'. It was created from a note in the repository, added labels, moved to 'In progress', and self-assigned. A recent comment from the author links to a document named 'Using_Git.md'.

Let's take a look at the file

Started document
Commit: 8d10d86
File: https://github.com/HaroonRafique/Accelerator_Simulation_Framework/blob/main/00_Getting_Started/Using_Git.md

Accelerator Simulation Framework: Git usage instructions

https://github.com/HaroonRafique/Accelerator_Simulation_Framework/blob/main/00_Getting_Started/Using_Git.md

The screenshot shows the GitHub repository page for 'HaroonRafique / Accelerator_Simulation_Framework'. The file 'Using_Git.md' is displayed, showing its content and metadata (165 lines, 5.02 KB). The page includes sections on 'Using Git: A guide to the basics', 'Using a web browser', and 'Basic Commands'.

Using Git: A guide to the basics

Basic Commands

How to edit this file (or similar)

- Navigate to the appropriate file: e.g. https://github.com/HaroonRafique/Accelerator_Simulation_Framework/blob/main/00_Getting_Started/Using_Git.md
- Ensure that you have permission to edit this file - if not, request permission, **fork** the repository to create your own copy, create your own **branch**, or create your own file
- Once open in a browser, in the top right corner of the file box locate the edit button (✍)
- Edit the file using markdown (<https://www.markdownguide.org/basic-syntax/>)
- Add emojis if you're feeling adventurous 😊 (<https://github.com/ikatyang/emoji-cheat-sheet#book-paper>)
- Navigate to the bottom of the page ↴
- In the Commit changes box, enter a commit message (*See style guide for tips on writing commit messages*), and press the green Commit changes button 🟩

The screenshot shows the 'Using Linux' section of the 'Using_Git.md' file. It includes a table of contents and several sections on basic Git commands with examples.

Using Linux

Basic Commands

From the command line

- **git config** Get and set repository or global options
- **git init** Create an empty Git repository or reinitialize an existing one
- **git clone** Clone a repository into a new directory
- **git add** Add file contents to Git index
- **git commit** Record changes to the repository
- **git diff** Show changes between commits, commit and working tree, etc
- **git status** Show the working tree status

Accelerator Simulation Framework: Git style guide

https://github.com/HaroonRafique/Accelerator_Simulation_Framework/blob/main/00_Getting_Started/Style_Guide.md

The screenshot shows the GitHub repository page for 'Accelerator_Simulation_Framework'. The file '00_Getting_Started/Style_Guide.md' is displayed, containing 53 lines and 2.09 KB in size. The content is organized into several sections:

- Commit messages**: A section explaining that committing changes to files requires a commit message to record the basics of said changes. It provides the command 'git commit -m 'commit message''.
- Git commit syntax**: A detailed explanation of the syntax. It states: "Currently we adopt the following syntax and format for a Git commit message in an Accelerator Simulation Framework repository:
<type>[main directories] additional information if necessary".
- Example sequence of commit messages**: A terminal session showing a sequence of commits: "git commit -m '<add>[hello world] initial code', git commit -m '<remove>[hello world] initial code removed', git commit -m '<bug>[my simulation] SCARF submission script not working', git commit -m '<bugfix>[my simulation] SCARF submission script typo corrected', git commit -m '<doc>[my simulation] instructions for correctly running on SCARF', git commit -m '<output>[my simulation] completed tracking sim on SCARF, bunch distributions added', git commit -m '<output>[my simulation] bunch plotting script and plots added', git commit -m '<example>[my example simulation] tested example of particle tracking dumping the bunch at every turn'".

The screenshot shows the GitHub repository page for 'Accelerator_Simulation_Framework'. The file '00_Getting_Started/Style_Guide.md' is displayed, containing 53 lines and 2.09 KB in size. The content is organized into several sections:

- Commit messages**: A section explaining that committing changes to files requires a commit message to record the basics of said changes. It provides the command 'git commit -m 'commit message''.
- Git commit syntax**: A detailed explanation of the syntax. It states: "In its complete form one would write the following in a terminal:
git commit -m '<type>[main directories] additional information'".
- Type**: Examples of types include: add, update, remove, bugfix, output, doc, example.
- Main Directories**: Examples of main directories include: directories/files changed.
- Additional Information**: Examples of additional information include: status (complete/incomplete/ongoing), details (bug, bugfix, add, update, output, remove, doc, example).
- Example sequence of commit messages**: A terminal session showing a sequence of commits: "git commit -m '<add>[hello world] initial code', git commit -m '<remove>[hello world] initial code removed', git commit -m '<bug>[my simulation] SCARF submission script not working', git commit -m '<bugfix>[my simulation] SCARF submission script typo corrected', git commit -m '<doc>[my simulation] instructions for correctly running on SCARF', git commit -m '<output>[my simulation] completed tracking sim on SCARF, bunch distributions added', git commit -m '<output>[my simulation] bunch plotting script and plots added', git commit -m '<example>[my example simulation] tested example of particle tracking dumping the bunch at every turn'".



Science and
Technology
Facilities Council

Opportunities

What could we do with a shared framework?



High Performance Computing (HPC)

Possible headaches:

- Installing your code on new HPC system
- New methods of running; queues, submission etc
- Optimisation: which queue is best, how many nodes...
- I've never done this and don't want to break it

Shared framework:

- One person installs their code for everyone to use
- One person provides example submission scripts
- Everyone can share experience on optimisation
- Less risk of doing something problematic /confusion



PyORBIT Examples

Created primarily for CERN Proton Synchrotron
and Booster space charge simulations on HPC

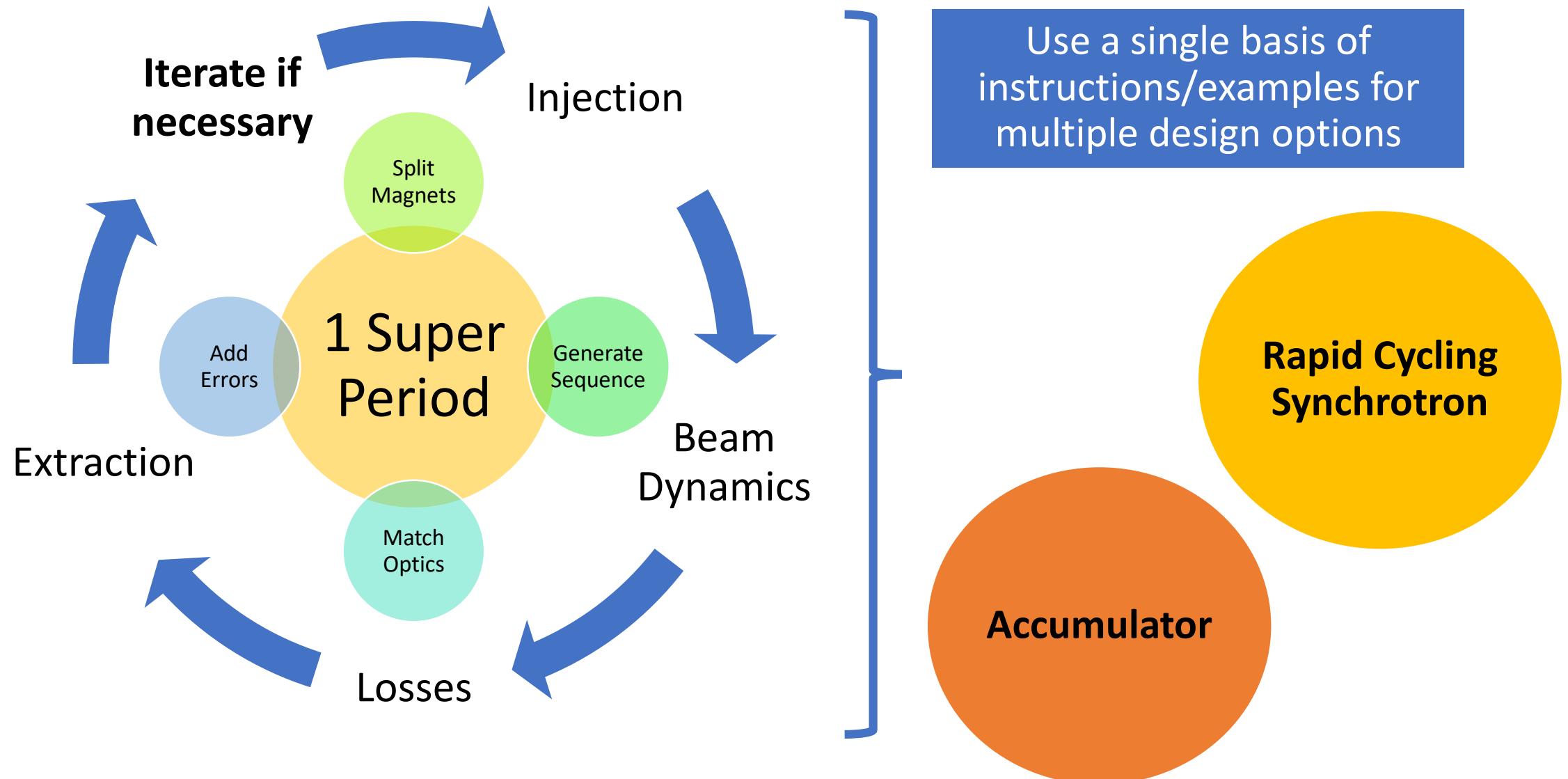
The screenshot shows the GitLab interface for the PyORBIT_Examples repository. At the top, there's a navigation bar with links to ISIS, UK SBS, CodIMD - Collabor..., ISIS-II, Teams, Sign In, and ISIS Intranet. Below this is a search bar and a 'Clone' button. The main area displays the 'master' branch with one commit: 'pyorbit_examples / PyORBIT_Examples' authored by PTC PyORBIT 2 years ago. An orange arrow points from this commit to the corresponding commit in the larger screenshot below.

Name	Last commit	Last update
..		
Create_A_Bunch	PyORBIT_Examples updated	2 years ago
Create_A_Lattice	Bugfixes	2 years ago
Distribution_from_tomo	Added PyORBIT_Examples from Haro...	2 years ago
Verify_Longitudinal_Motion	Added tomo image to Verify_Lonitu...	2 years ago

This screenshot shows the full commit history and file structure for the PyORBIT_Examples repository. At the top, it shows basic project statistics: 57 Commits, 1 Branch, 0 Tags, 160.6 MB Files, and 160.6 MB Storage. Below this is a summary message: 'Added ACCSSIM user guides (predecessor to ORBIT)' by PTC PyORBIT 1 year ago. A blue box highlights the 'PyORBIT_Examples' commit. The main area is a table of commits:

Name	Last commit	Last update
AFS_Phaseout_Test	AFS_Phaseout_Test: Renamed and up...	1 year ago
Job_Submission	Minor updates added	1 year ago
Machines	Minor updates added	1 year ago
PTC_Scripts	Added PTC_Scripts: General/ contain...	2 years ago
PyORBIT_Examples	Bugfixes	2 years ago
PyOrbit_documentation	Added ACCSSIM user guides (predec...	1 year ago
lib	Added lib/ to hold PyORBIT library fil...	2 years ago
.gitignore	correct bug in plot_output.py	1 year ago

ISIS-II Accelerator Design Repository: Ideas



Long Term (Code) Development: An example

github.com/Merlin-Collaboration Oct 11, 2009 – May 10, 2021 Contributions: Commits

Search or jump to... Pull requests Issues Marketplace Explore

Merlin++
Merlin++ is a multi-purpose particle accelerator and particle tracking simulation library
<http://merlinpp.org/>

Repositories 2 Packages People 4 Teams Projects Settings

Find a repository... Type Language Sort Customize pins

Merlin
Merlin++ Particle Accelerator Simulation Library

tracking simulation fcc particle particle-accelerator lhcb merlin

C++ GPL-2.0 9 8 0 1 Updated 7 days ago

Nightly-build-scripts

CMake 2 0 1 0 Updated on 31 Jul 2020

Top languages C++ CMake

People

Invite someone

JamesMolson 303 commits 653,134 ++ 337,721 -- #1

Sam-Tygier 258 commits 54,287 ++ 59,463 -- #2

HaroonRafique 53 commits 17,503 ++ 8,973 -- #3

scottmrowan 26 commits 999,076 ++ 962,984 -- #4

jfallon1997 17 commits 6,139 ++ 3,478 -- #5

RogerJBarlow 4 commits 87 ++ 57 -- #6

<https://github.com/Merlin-Collaboration/Merlin/network>
<https://github.com/HaroonRafique/MERLIN/network>

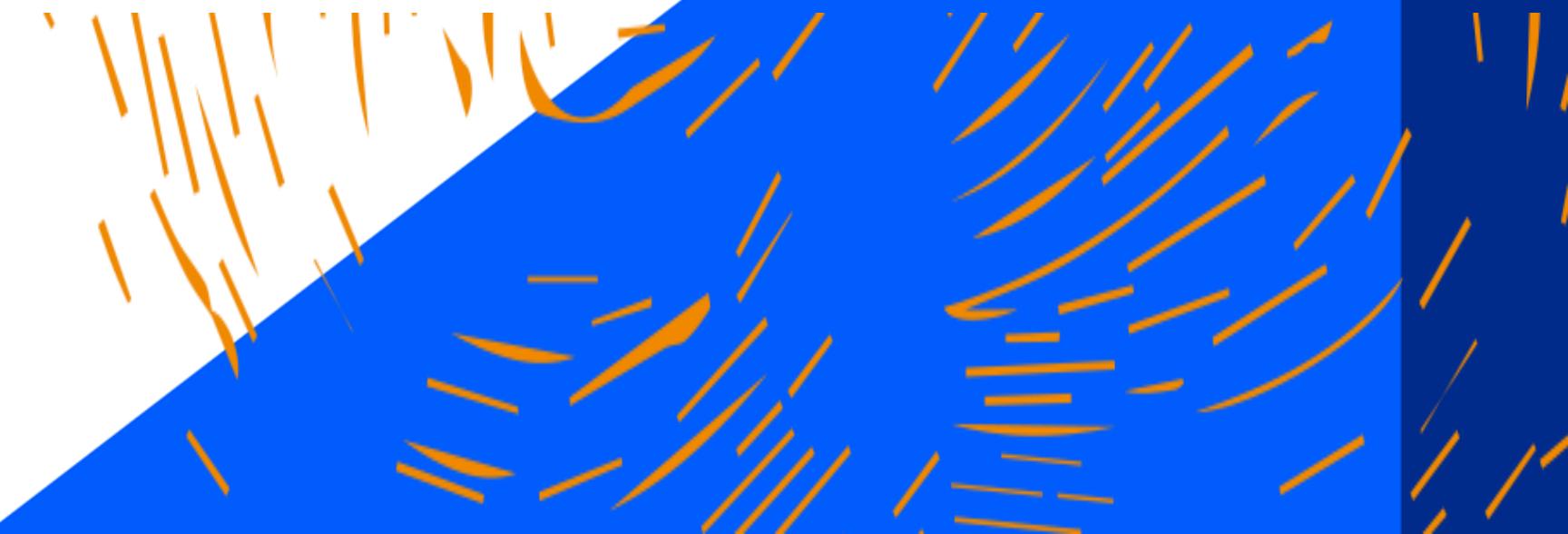
UK RI



Science and
Technology
Facilities Council

Debate

Working Together:
Problems? Ideas? Questions?



A Shared Framework Should Make Working Easier

If your code/simulation is large/complex or otherwise un-git-able

- Store input files etc required to reproduce important simulations
- Save (small) output files and plotting scripts to ease the write-up process (changing plots months after simulation is easy)

If your code/simulation is not useful to others

- Your work may be useful in the future, even in parts, and can save time as a reference at minimum
- Using a repository is a safe way of organising code, and can be useful when revisiting things years later

The time it would take to put it onto GitLab/GitHub is similar to the work itself

- Git becomes easier with use, and if we all use it as a default then this should save time in the long term
- Spending time to future-proof and make your hard work accessible should benefit the work of the group
- It is up to you to decide how best to spend your time 😊

Most of my work is messy

- Keep your messy work in your own branch/fork or even make your own repository/repositories that are more private and not designed for others to use
- Only contribute tested, well documented work to shared repositories/projects

Issues not solved by Git

Large file storage

- Git has a file limit of 50 megabytes
- Can we use **IDAaaS** / neutron user infrastructure
- Should we use **internal file storage?**

Overcoming the fear of change

- **Take a leap of faith?**
- Things are **likely to become more complex** with time (with regards to computing infrastructure, codes, tools, languages)
- Any problem is an opportunity to improve how things are done
- You **can't do everything on windows** (sorry-not-sorry STFC)
- I have **supported user codes / git repositories / HPC installations** for years – **happy to help** with anything
- **Don't know how to do something?** No matter how insignificant/stupid it may seem – **opportunity to contribute**
 - Put it in the README/documentation/ideas/suggestions/issues/bugs
 - Maybe ask Prof. Google

Remember

- **Time saving, future proofing, collaborating, pushing simulation complexity, exploiting HPC, bug-fixing, idiot-proofing, pandemic-proofing, ...**



Science and
Technology
Facilities Council

Thank you

haroon.rafique@stfc.ac.uk