Global analyses of supersymmetry with GAMBIT

Pat Scott

Imperial College
London

on behalf of the GAMBIT Collaboration

Slides at: www.imperial.ac.uk/people/p.scott/research.html
GAMBIT: gambit.hepforge.org
Even the simplest particle theories have *many* different consequences...theoretical, experimental, computational...
Even the simplest particle theories have *many* different consequences...theoretical, experimental, computational... 
Global fits = let’s try to capture, control and satisfy them all.
Global fits for new physics

Even the simplest particle theories have many different consequences...theoretical, experimental, computational...

Global fits = let’s try to capture, control and satisfy them all.

Current global fit codes are hardcoded to deal with only a few

- theories (MSSM and/or mSUGRA+friends)
- theory calculators (often interfaced in a very ad hoc way)
- datasets and observables (often missing detailed likelihoods)
- scanning algorithms and statistical methods (generally just one)

⇒ hitting the wall on theories, data & computational methods
Global fits for new physics

Even the simplest particle theories have *many* different consequences...theoretical, experimental, computational... Global fits = let’s try to capture, control and satisfy them all.

Current global fit codes are hardcoded to deal with only a few

- theories (MSSM and/or mSUGRA+friends)
- theory calculators (often interfaced in a very ad hoc way)
- datasets and observables (often missing detailed likelihoods)
- scanning algorithms and statistical methods (generally just one)

⇒ *hitting the wall on theories, data & computational methods*

How to quickly recast data, likelihood functions, scanning code ‘housekeeping’ and even theory predictions to new theories?
⇒ a new, very general global fitting framework
Even the simplest particle theories have *many* different consequences...theoretical, experimental, computational... 

**Global fits = let’s try to capture, control and satisfy them all.**

Current global fit codes are hardcoded to deal with only a few

- theories (MSSM and/or mSUGRA+friends)
- theory calculators (often interfaced in a very ad hoc way)
- datasets and observables (often missing detailed likelihoods)
- scanning algorithms and statistical methods (generally just one)

⇒ *hitting the wall on theories, data & computational methods*

How to quickly recast data, likelihood functions, scanning code ‘housekeeping’ and even theory predictions to new theories?

⇒ a new, very general global fitting framework

⇒ **GAMBIT**
GAMBIT: The Global And Modular BSM Inference Tool


- Fast definition of new datasets and theoretical models
- Plug and play scanning, physics and likelihood packages
- Extensive model database – not just SUSY
- Extensive observable/data libraries
- Many statistical and scanning options (Bayesian & frequentist)
- Fast LHC likelihood calculator
- Massively parallel
- Fully open-source

ATLAS
F. Bernlochner, A. Buckley, P. Jackson, M. White
LHCb
M. Chrząszcz, N. Serra
Belle-II
F. Bernlochner, P. Jackson
Fermi-LAT
J. Conrad, J. Edsjö, G. Martinez, P. Scott
CTA
C. Balázs, T. Bringmann, M. White
CMS
C. Rogan
IceCube
J. Edsjö, P. Scott
XENON/DARWIN
B. Farmer, R. Trolta
Theory
P. Athron, C. Balázs, S. Bloor, T. Bringmann,
J. Cornell, J. Edsjö, B. Farmer, A. Fowlie, T. Gonzalo,
J. Harz, S. Hoof, F. Kahlhoefer, S. Krishnamurthy,
A. Kvellestad, F.N. Mahmoudi, J. McKay, A. Raklev,
R. Ruiz, P. Scott, R. Trolta, A. Vincent, C. Weniger,
M. White, S. Wild

31 Members in 9 Experiments, 12 major theory codes, 11 countries
Physics modules

- **DarkBit** – dark matter observables (relic density, direct + indirect detection)  
  (EPJC, arXiv:1705.07920)

- **ColliderBit** – collider observables inc. Higgs + SUSY searches from ATLAS, CMS + LEP  
  (EPJC, arXiv:1705.07919)

- **FlavBit** – flavour physics inc. $g - 2$, $b \to s\gamma$, $B$ decays (new channels, angular obs., theory uncerts, LHCb likelihoods)  
  (EPJC, arXiv:1705.07933)

- **SpecBit** – generic BSM spectrum object, providing RGE running, masses, mixings, etc via interchangeable interfaces to different RGE codes  
  (EPJC, arXiv:1705.07936)

- **DecayBit** – decay widths for all relevant SM & BSM particles  
  (EPJC, arXiv:1705.07936)

- **PrecisionBit** – SM likelihoods, precision BSM tests ($W$ mass, $\Delta \rho$ etc)  
  (EPJC, arXiv:1705.07936)

Each consists of a number of module functions that can have dependencies on each other.
Models live in a hierarchy in GAMBIT:

This provides many neat computational advantages – quiz me at coffee/dinner.
GUT-scale MSSM (CMSSM) (EPJC, arXiv:1705.07935)

Global analyses of supersymmetry with GAMBIT

$m_0, m_{1/2}, A_0, \tan \beta + 5$ nuisances

$H/A^0$ funnel, $\chi^\pm$ co-annihilation, $\tilde{t}$ co-annihilation

$\tilde{\tau}$ co-annihilation now ruled out

Includes LUX, Panda-X, Fermi, flavour + direct sim. of ATLAS & CMS Run 1 & Run 2 limits.
GUT-scale MSSM (CMSSM) (EPJC, arXiv:1705.07935)

GAMBIT 1.0.0

CMSSM

Best fit

$A_0$ (GeV)

$\tan \beta$

Profile likelihood ratio $\Lambda = \frac{L}{L_{\text{max}}}$

$0.2$

$0.4$

$0.6$

$0.8$

$1.0$

$m_0, m_{1/2}, A_0, \tan \beta + 5$ nuisances

$H/A^0$ funnel, $\chi^\pm$ co-annihilation, $\tilde{t}$ co-annihilation

$\tilde{\tau}$ co-annihilation now ruled out

Includes LUX, Panda-X, Fermi, flavour + direct sim. of ATLAS & CMS Run 1 & Run 2 limits.
GUT-scale MSSM (CMSSM)

$\Omega h^2 = 0.119 \pm 0.025 \pm 0.020 \pm 0.015 \pm 0.010$

$\log_{10}(\Omega \chi h^2)$

Profile likelihood ratio $\Lambda = \frac{L}{L_{\text{max}}}$

$m_{\tilde{t}_1}$ co-annihilation
$A/H$ funnel
$\tilde{\chi}_1^\pm$ co-annihilation

- $m_0$, $m_{\tilde{t}}$, $A_0$, $\tan \beta$ + 5 nuisances
- $H/A^0$ funnel, $\chi^\pm$ co-annihilation, $\tilde{t}$ co-annihilation
- $\tilde{\tau}$ co-annihilation now ruled out
- Includes LUX, Panda-X, Fermi, flavour + direct sim. of ATLAS & CMS Run 1 & Run 2 limits.
GUT-scale MSSM (CMSSM)

- $m_0$, $m_1^2$, $A_0$, $\tan \beta$ + 5 nuisances
- $H/A^0$ funnel, $\chi^\pm$ co-annihilation, $\tilde{t}$ co-annihilation
- $\tilde{t}$ co-annihilation now ruled out
- $\tilde{t}$ co-annihilation tough to get at with DD, ID or LHC – but some hope with LHC for seeing light stops at the ‘tip’
**GUT-scale MSSM (NUHM1/2)**

(EPJC, arXiv:1705.07935)

---

**Neutrino floor**

**NUHM1**

**NUHM2**

\[
\log_{10}(f \cdot \sigma_{SI}/\text{cm}^2)
\]

\[
m_{\tilde{\chi}_1^0} (\text{GeV})
\]

- \(\tilde{t}_1\) co-annihilation
- \(A/H\) funnel
- \(\tilde{\chi}_1^\pm\) co-annihilation
- \(\tilde{\tau}_1\) co-annihilation

- \(m_0, m_H (m_{H_u}, m_{H_d}), m_{\frac{1}{2}}, A_0, \tan \beta + 5\) nuisances

- \(H/A^0\) funnel, \(\chi^\pm\) co-annihilation, \(\tilde{t}\) co-annihilation + \(\tilde{\tau}\) co-annihilation due to additional freedom to fit \(m_h\)

- Includes LUX, Panda-X, Fermi, flavour + direct sim. of ATLAS & CMS Run 1 & Run 2 limits.

---

**Pat Scott – Mar 20 2018 – Moriond QCD**

Global analyses of supersymmetry with GAMBIT
Global analyses of supersymmetry with GAMBIT

Weak-scale MSSM (MSSM-7) (EPJC, arXiv:1705.07917)

- $\tilde{t}_1$ co-annihilation
- $A/H$ funnel
- $\tilde{\chi}^\pm_1$ co-annihilation
- $\tilde{b}_1$ co-annihilation
- $h/Z$ funnel

- $m_{\tilde{t}}, M_2, A_u, A_d, m_{H_u}, m_{H_d}, \tan \beta + 5$ nuisances
- $H/A^0$ funnel, $h/Z$ funnel, $\chi^\pm$ co-annihilation, $\tilde{t}/\tilde{b}$ co-annihilation

- Includes LUX, Panda-X, Fermi + direct sim. of LHC Run 1 & Run 2 limits.

MSSM-9 coming later this year (including more Run 2 searches)
Weak-scale MSSM (MSSM-7)

Global analyses of supersymmetry with GAMBIT
Global analyses of supersymmetry with GAMBIT

Weak-scale MSSM (MSSM-7)

MSSM7

Best fit

LUX 2016
XENON1T (2ty)
XENONnT/LZ (20ty)
DARWIN (200ty)

Neutrino floor

\[
\log_{10}(f \cdot \sigma_{SI}/\text{cm}^2)
\]

Profile likelihood ratio \( \Lambda = L/L_{\text{max}} \)

\[0, 1000, 2000, 3000\]

\[m_{\tilde{\chi}_0^1} \text{ (GeV)}\]

\[0.2, 0.4, 0.6, 0.8, 1.0\]

\[m_{\tilde{t}_1}, M_2, A_u, A_d, m_{H_u}, m_{H_d}, \tan \beta + 5 \text{ nuisances}\]

\[H/A^0 \text{ funnel, } h/Z \text{ funnel, } \chi^\pm \text{ co-annihilation, } \tilde{t}/\tilde{b} \text{ co-annihilation}\]

Includes LUX, Panda-X, Fermi + direct sim. of LHC Run 1 & Run 2 limits.

MSSM-9 coming later this year (including more Run 2 searches)
Global analyses of supersymmetry with GAMBIT

Weak-scale MSSM (MSSM-7)

- $m_{\tilde{f}}$, $M_2$, $A_u$, $A_d$, $m_{H_u}$, $m_{H_d}$, $\tan \beta$ + 5 nuisances
- $H/A^0$ funnel, $h/Z$ funnel, $\chi^{\pm}$ co-annihilation, $\tilde{t}/\tilde{b}$ co-annihilation
- Includes LUX, Panda-X, Fermi + direct sim. of LHC Run 1 & Run 2 limits.

MSSM-9 coming later this year (including more Run 2 searches)
Other models (a taste)
(EPJC, EPJC, arXiv:1705.07931 & in prep)

Scalar singlet DM ($\mathbb{Z}_2$ & $\mathbb{Z}_3$)

Higgs Portal DM

Axions & ALPs

- flavour EFTs
- light neutralinos & charginos
- right-handed neutrinos
- 2HDMs
- ...
Summary

- Stop co-annihilation generally looking much more promising than previously thought
- Stau co-annihilation is out for the CMSSM
- Pretty much anything is still possible within the weak-scale (p)MSSM
- Direct detection probably the most promising future probe

- Global analyses of many other models already available or coming soon
- GAMBIT results, samples, run files, best fits, benchmarks, etc are all available to download from Zenodo: www.zenodo.org/communities/gambit-official/
- GAMBIT code is public: gambit.hepforge.org
Backup slides
Scalar singlet DM \((m_S, \lambda_{hS} + 13\) nuisances\) (EPJC, arXiv:1705.07931)

Profile likelihood ratio \(\Lambda = \frac{L}{L_{\text{max}}}
\)

Simplest BSM example:
\[
\mathcal{L}_S = -\frac{\mu_S^2}{2} S^2 - \frac{\lambda_{hS}}{2} S^2 H^\dagger H + \ldots
\]

All dark matter signals consistently scaled for predicted abundance.
Scalar singlet DM ($m_S$, $\lambda h_S + 13$ nuisances) 

(EPJC, arXiv:1705.07931)

Simplest BSM example: $\mathcal{L}_S = -\frac{\mu_S^2}{2} S^2 - \frac{\lambda_{hs}}{2} S^2 H^\dagger H + \ldots$

All dark matter signals consistently scaled for predicted abundance
Scalar singlet DM \((m_S, \lambda_{hS} + 13\) nuisances\)

\[ \Omega h^2 = 0.119 \]

Simplest BSM example:

\[ \mathcal{L}_S = -\frac{\mu_S^2}{2} S^2 - \frac{\lambda_{hS}}{2} S^2 H^\dagger H + \ldots \]

All dark matter signals consistently scaled for predicted abundance.
Scalar singlet DM ($m_S$, $\lambda_{hS} + 13$ nuisances)  

Simplest BSM example: $\mathcal{L}_S = -\frac{\mu_S^2}{2} S^2 - \frac{\lambda_{hS}}{2} S^2 H^\dagger H + \ldots$

All dark matter signals consistently scaled for predicted abundance.

Global analyses of supersymmetry with GAMBIT
Scalar singlet DM ($m_S$, $\lambda_{hS} + 13$ nuisances)

(EPJC, arXiv:1705.07931)
Global analyses of supersymmetry with GAMBIT
• Models are defined by their parameters and relations to each other
• Models can inherit from (be subspaces of) **parent models**
• Points in child models can be **automatically translated** to ancestor models
• **Friend models** also allowed (cross-family translation)
• Model dependence of every function/observable is tracked

⇒ maximum safety, maximum reuse
Module functions can require specific functions from backends.

Backends are external code libraries (DarkSUSY, FeynHiggs, etc) that include different functions.

GAMBIT automates and abstracts the interfaces to backends → backend functions are tagged according to what they calculate.

→ with appropriate module design, different backends and their functions can be used interchangeably.

GAMBIT dynamically adapts to use whichever backends are actually present on a user’s system (+ provides details of what it decided to do of course).
Global analyses of supersymmetry with GAMBIT

Backends: mix and match

(EPJC, arXiv:1705.07908)

Pat Scott – Mar 20 2018 – Moriond QCD

---

<table>
<thead>
<tr>
<th>BACKENDS</th>
<th>VERSION</th>
<th>PATH TO LIB</th>
<th>STATUS</th>
<th>#FUNC</th>
<th>#TYPES</th>
<th>#CTORS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDCalc0</td>
<td>0.9</td>
<td>Backends/installed/DDCalc/0.8/libDDCalc0.so</td>
<td>OK</td>
<td>62</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DarkSUSY</td>
<td>5.1.1</td>
<td>Backends/installed/DarkSUSY/3.1.1/lib/libdarksusy.so</td>
<td>OK</td>
<td>68</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FastSim</td>
<td>1.0</td>
<td>Backends/installed/fastsim/1.0/libfastsim.so</td>
<td>absent/broken</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FeynHiggs</td>
<td>2.11</td>
<td>Backends/installed/FeynHiggs/2.11.2/lib/libFH.so</td>
<td>OK</td>
<td>14</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HiggsBounds</td>
<td>4.2.1</td>
<td>Backends/installed/HiggsBounds/4.2.1/lib/libhiggsbounds.so</td>
<td>OK</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HiggsSignals</td>
<td>1.4</td>
<td>Backends/installed/HiggsSignals/1.4.0/lib/libhiggssignals.so</td>
<td>OK</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LibFarrayTest</td>
<td>1.0</td>
<td>Backends/examples/libFarrayTest.so</td>
<td>OK</td>
<td>9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LibFirst</td>
<td>1.0</td>
<td>Backends/examples/libfirst.so</td>
<td>OK</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1.1</td>
<td>Backends/examples/libfirst.so</td>
<td>OK</td>
<td>15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LibFortran</td>
<td>1.0</td>
<td>Backends/examples/libfortran.so</td>
<td>OK</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MicrOmesgas</td>
<td>3.5.5</td>
<td>Backends/installed/micromegas/3.5.5/MSSM/MSSM/libmicromegas.so</td>
<td>OK</td>
<td>15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MicrOmesgasSingletDM</td>
<td>3.5.5</td>
<td>Backends/installed/micromegas/3.5.5/SingletDM/SingletDM/libmicromegas.so</td>
<td>OK</td>
<td>13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Pythia</td>
<td>8.186</td>
<td>Backends/installed/Pythia/8.186/lib/libpythia8.so</td>
<td>absent/broken</td>
<td>0</td>
<td>27</td>
<td>105</td>
</tr>
<tr>
<td></td>
<td>8.209</td>
<td>Backends/installed/Pythia/8.209/lib/libpythia8.so</td>
<td>OK</td>
<td>0</td>
<td>28</td>
<td>107</td>
</tr>
<tr>
<td>SUSYPOPE</td>
<td>0.2</td>
<td>no path in config/backend_locations.yaml</td>
<td>absent/broken</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SUSY_HIT</td>
<td>1.5</td>
<td>Backends/installed/SUSY_HIT/1.5/libsusyhit.so</td>
<td>OK</td>
<td>55</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SuperIso</td>
<td>3.4</td>
<td>Backends/installed/SuperIso/3.4/libsuperiso.so</td>
<td>OK</td>
<td>32</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>gamLike</td>
<td>1.0.0</td>
<td>Backends/installed/gamLike/1.0.0/lib/gamLike.so</td>
<td>OK</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nulike</td>
<td>1.0.0</td>
<td>Backends/installed/nulike/1.0.0/lib/libnulike.so</td>
<td>OK</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Gambit diagnostic backend line 1 (press h for help or q to quit)
Backends: mix and match

Global analyses of supersymmetry with GAMBIT

<table>
<thead>
<tr>
<th>BACKENDS</th>
<th>VERSION</th>
<th>PATH TO Lib</th>
<th>STATUS</th>
<th>#FUNC</th>
<th>#TYPES</th>
<th>#CTORS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDCalc0</td>
<td>0.9</td>
<td>Backends/installed/DDCalc/0.8/libDDCalc0.so</td>
<td>OK</td>
<td>62</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Dark SUSY</td>
<td>5.1.1</td>
<td>Backends/installed/DarkSUSY/3.1.1/lib/libdarksusy.so</td>
<td>OK</td>
<td>68</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FastSim</td>
<td>1.0</td>
<td>Backends/installed/fastsim/1.0/libfastsim.so</td>
<td>absent/broken</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FeynHiggs</td>
<td>2.11</td>
<td>Backends/installed/FeynHiggs/2.11.2/lib/libFHi.so</td>
<td>OK</td>
<td>14</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HiggsBounds</td>
<td>4.2.1</td>
<td>Backends/installed/HiggsBounds/4.2.1/lib/libhiggsbounds.so</td>
<td>OK</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HiggsSignals</td>
<td>1.4</td>
<td>Backends/installed/HiggsSignals/1.4.0/lib/libhiggssignals.so</td>
<td>OK</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LibFarrayTest</td>
<td>1.0</td>
<td>Backends/examples/libFarrayTest.so</td>
<td>OK</td>
<td>9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LibFirst</td>
<td>1.0</td>
<td>Backends/examples/libfirst.so</td>
<td>OK</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1.1</td>
<td>Backends/examples/libfirst.so</td>
<td>OK</td>
<td>15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LibFortran</td>
<td>1.0</td>
<td>Backends/examples/libfortran.so</td>
<td>OK</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MicrOmeas</td>
<td>3.5.5</td>
<td>Backends/installed/micrOmeas/3.5.5/MSSM/MSSM/libmicrOmeas.so</td>
<td>OK</td>
<td>15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MicrOmeasSingletDM</td>
<td>3.5.5</td>
<td>Backends/installed/micrOmeas/3.5.5/SingletDM/SingletDM/libmicrOmeas.so</td>
<td>OK</td>
<td>13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Pythia</td>
<td>8.186</td>
<td>Backends/installed/Pythia/8.186/lib/libpythia8.so</td>
<td>absent/broken</td>
<td>0</td>
<td>27</td>
<td>105</td>
</tr>
<tr>
<td></td>
<td>8.209</td>
<td>Backends/installed/Pythia/8.209/lib/libpythia8.so</td>
<td>OK</td>
<td>0</td>
<td>28</td>
<td>107</td>
</tr>
<tr>
<td>SUSYPOPE</td>
<td>0.2</td>
<td>no path in config/backend_locations.yaml</td>
<td>absent/broken</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SUSY_HIT</td>
<td>1.5</td>
<td>Backends/installed/SUSY-HIT/1.5/libSUSYHit.so</td>
<td>OK</td>
<td>55</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SuperIso</td>
<td>3.4</td>
<td>Backends/installed/SuperIso/3.4/libSuperIso.so</td>
<td>OK</td>
<td>32</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>gamLike</td>
<td>1.0.0</td>
<td>Backends/installed/gamLike/1.0.0/lib/gamLike.so</td>
<td>OK</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nulike</td>
<td>1.0.0</td>
<td>Backends/installed/nulike/1.0.0/lib/libnulike.so</td>
<td>OK</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Gambit diagnostic backend line 1 (press h for help or q to quit)
• Module functions and backend functions get arranged into a **dependency tree**
• Starting with requested observables and likelihoods, GAMBIT fills each dependency and backend requirement
• Obeys **rules** at each step: allowed models, allowed backends, constraints from input file, etc
• → tree constitutes a directed acyclic graph
• → GAMBIT uses graph-theoretic methods to ‘solve’ the graph to determine function evaluation order
CMSSM:

MSSM7:
CMSSM:

MSSM7:

Red: Model parameter translations
CMSSM:

MSSM7:

Red: Model parameter translations
Blue: Precision calculations
CMSSM:

MSSM7:

Red: Model parameter translations
Blue: Precision calculations
Green: LEP rates + likelihoods
CMSSM:

MSSM7:

Red: Model parameter translations
Blue: Precision calculations
Green: LEP rates + likelihoods
Purple: Decays
Red: Model parameter translations
Blue: Precision calculations
Green: LEP rates + likelihoods
Purple: Decays
Orange: LHC observables and likelihoods
CMSSM:

Red: Model parameter translations
Blue: Precision calculations
Green: LEP rates + likelihoods
Purple: Decays
Orange: LHC observables and likelihoods
Grey: DM direct, indirect and relic density

MSSM7:
CMSSM:

MSSM7:

Red: Model parameter translations
Blue: Precision calculations
Green: LEP rates+likelihoods
Purple: Decays
Orange: LHC observables and likelihoods
Grey: DM direct, indirect and relic density
Pink: Flavour physics
Extensive scanner tests on scalar singlet model with different numbers of nuisance parameters

Diver scales far better with dimensionality than MultiNest or other scanners
Adding a new module function is easy:

1. Declare the function to GAMBIT in a module’s rollcall header

   - Choose a capability
   - Declare any **backend requirements**
   - Declare any **dependencies**
   - Declare any specific **allowed models**
   - other more advanced declarations also available

```c
#define MODULE FlavBit // A tasty GAMBIT module.
START_MODULE

#define CAPABILITY Rmu // Observable: BR(K->mu nu)/BR(pi->mu nu)
START_CAPABILITY
#define FUNCTION SI_Rmu // Name of a function that can compute Rmu
START_FUNCTION(double) // Function computes a double precision result
   BACKEND_REQ(Kmunu_pimunu, (my_tag), double, (const parameters*)) // Needs function from a backend
   BACKEND_OPTION( (SuperIso, 3.6), (my_tag) ) // Backend must be SuperIso 3.6
   DEPENDENCY(SuperIso_modelinfo, parameters) // Needs another function to calculate SuperIso info
   ALLOW_MODELS(MSSM63atQ, MSSM63atMGUT) // Works with weak/GUT-scale MSSM and descendents
#undef FUNCTION
#undef CAPABILITY
```

2. Write the function as a standard C++ function
   (one argument: the result)
Expansion: adding new models

1. Add the model to the **model hierarchy**:
   - Choose a model name, and declare any **parent model**
   - Declare the model’s parameters
   - Declare any **translation function** to the parent model

```cpp
#define MODEL NUHM1
#define PARENT NUHM2

START_MODEL
  DEFINEPARS(M0,M12,mH,A0,TanBeta,SignMu)
  INTERPRET_AS_PARENT_FUNCTION(NUHM1_to_NUHM2)
#undef PARENT
#undef MODEL
```

2. Write the translation function as a standard C++ function:

```cpp
void MODEL_NAMESPACE::NUHM1_to_NUHM2 (const ModelParameters &myP, ModelParameters &targetP)
{
  // Set M0, M12, A0, TanBeta and SignMu in the NUHM2 to the same values as in the NUHM1
  targetP.setValues(myP,false);
  // Set the values of mHu and mHd in the NUHM2 to the value of mH in the NUHM1
  targetP.setValue("mHu", myP["mH"]);
  targetP.setValue("mHd", myP["mH"]);
}
```

3. If needed, declare that existing module functions work with the new model, or add new functions that do.
Basic interface for a scan is a YAML initialisation file

- specify parameters, ranges, priors
- select likelihood components
- select other observables to calculate
- define generic rules for how to fill dependencies
- define generic rules for options to be passed to module functions
- set global options (scanner, errors/warnings, logging behaviour, etc)
Basic interface for a scan is a YAML initialisation file

- specify parameters, ranges, priors
- select likelihood components
- select other observables to calculate
- define generic rules for how to fill dependencies
- define generic rules for options to be passed to module functions
- set global options (scanner, errors/warnings, logging behaviour, etc)
Basic interface for a scan is a YAML initialisation file

- specify parameters, ranges, priors
- select likelihood components
- select other observables to calculate
- define generic rules for how to fill dependencies
- define generic rules for options to be passed to module functions
- set global options (scanner, errors/warnings, logging behaviour, etc)
Basic interface for a scan is a YAML initialisation file

- specify parameters, ranges, priors
- select likelihood components
- select other observables to calculate
- define generic rules for how to fill dependencies
- define generic rules for options to be passed to module functions
- set global options (scanner, errors/warnings, logging behaviour, etc)
Basic interface for a scan is a YAML initialisation file

- specify parameters, ranges, priors
- select likelihood components
- select other observables to calculate
- define generic rules for how to fill dependencies
- define generic rules for options to be passed to module functions
- set global options (scanner, errors/warnings, logging behaviour, etc)
Other nice technical features

- **Scanners**: Nested sampling, differential evolution, MCMC, t-walk...
- Mixed-mode **MPI + openMP** parallelisation, mostly automated → scales to 10k+ cores
- **diskless generalisation of various Les Houches Accords**
- **BOSS**: dynamic loading of C++ classes from backends (!)
- **all-in or module standalone** modes – easily implemented from single cmake script
- **automatic getters** for obtaining, configuring + compiling backends\(^1\)
- **flexible output streams** (ASCII, databases, HDF5, . . .)
- available as docker plugin or vagrant virtual machine
- more more more more.

---

\(^1\)if a backend won’t compile/crashes/kills your cat, blame the authors (not us...except where we are the authors...)
ColliderBit details

LEP likelihoods

- complete model-independent recast of direct sparticle searches

Higgs likelihoods:

- for now: HiggSignals + HiggsBounds + constraints from invisible fits (Bernon, Dumont, Kraml et al)
- future: full simulation and ATLAS+CMS combination, more correlations, CP info, no SM-like coupling assumptions

Fast LHC likelihoods

- no simplified models, just faster direct simulation
ColliderBit details

LHC likelihoods:

- **MC generation**: Pythia8 parallelised with OpenMP + other speed tweaks
ColliderBit details

LHC likelihoods:
- **MC generation**: Pythia8 parallelised with OpenMP + other speed tweaks
- **Detector simulation**: fast simulation based on 4-vector smearing
  → matches DELPHES results very closely (but much faster!)

Leading electron $p_T$ distribution (CMSSM example):
- **red**: detector-level simulation with DELPHES
- **green**: 4-vector smearing with GAMBIT
- **blue**: truth-level distribution
ColliderBit details

LHC likelihoods:

- **MC generation**: Pythia8 parallelised with OpenMP + other speed tweaks
- **Detector simulation**: fast simulation based on 4-vector smearing → matches DELPHES results very closely (but much faster!)
- **Cross-sections**: LO + LL from MC generator by default (fast NLO in works for SUSY)
LHC likelihoods:

- **MC generation**: Pythia8 parallelised with OpenMP + other speed tweaks
- **Detector simulation**: fast simulation based on 4-vector smearing → matches DELPHES results very closely (but much faster!)
- **Cross-sections**: LO + LL from MC generator by default (fast NLO in works for SUSY)
- **Analysis framework**: custom event-level, independent of experiment or simulation
ColliderBit details

LHC likelihoods:

- **MC generation**: Pythia8 parallelised with OpenMP + other speed tweaks
- **Detector simulation**: fast simulation based on 4-vector smearing → matches DELPHES results very closely (but much faster!)
- **Cross-sections**: LO + LL from MC generator by default (fast NLO in works for SUSY)
- **Analysis framework**: custom event-level, independent of experiment or simulation
- **Likelihood**: inline systematic error marginalisation (via nulike)

Initially shipping with:
- ATLAS SUSY searches (0ℓ, 0/1/2ℓ˜t, b jets + MET, 2/3ℓ EW)
- CMS multi-ℓ SUSY
- CMS DM (t pair + MET, mono-b, monojet)
- ATLAS + CMS Run II 0ℓ refresh coming with more Run II analyses shortly
ColliderBit details

LHC likelihoods:

- **MC generation**: Pythia8 parallelised with OpenMP + other speed tweaks
- **Detector simulation**: fast simulation based on 4-vector smearing
  → matches DELPHES results very closely (but much faster!)
- **Cross-sections**: LO + LL from MC generator by default (fast NLO in works for SUSY)
- **Analysis framework**: custom event-level, independent of experiment or simulation
- **Likelihood**: inline systematic error marginalisation (via nulike)
- **Initially shipping with**:
  - ATLAS SUSY searches (0\(\ell\), 0/1/2\(\ell\) \(\tilde{t}\), b jets + MET, 2/3\(\ell\) EW)
  - CMS multi-\(\ell\) SUSY
  - CMS DM (\(t\) pair + MET, mono-b, monojet)
  - ATLAS + CMS Run II 0\(\ell\)
  - refresh coming with more Run II analyses shortly
LHCb sees possible hints of lepton flavour non-universality in neutral currents → GAMBIT flavour EFT global fit (Wilson coefficients as model parameters)

Flavour likelihoods in GAMBIT:

\[(g - 2)_\mu\]
\[B \to X_s \gamma\]
\[B \to \mu\mu\]
\[B_s \to \mu\mu\]
\[B \to K^* \mu\mu + \text{angular observables}\]
\[B \to \tau\nu\]
\[B \to D\mu\nu\]
\[B \to D\tau\nu\]
\[B \to D^* \mu\nu\]
\[B \to D^* \tau\nu\]
\[D \to \mu\nu\]
\[D_s \to \mu\nu\]
\[D_s \to \tau\nu\]
\[\frac{B(K \to \mu\nu)}{B(\pi \to \mu\nu)}\]

(EPJC, arXiv:1705.07933)

Fit to \(\mathcal{O}_7\) (photons), \(\mathcal{O}_9\) (leptons, vector), \(\mathcal{O}_{10}\) (leptons, axial-vector)

GAMBIT is also an official part of the Belle II Theory Interface Platform (B2TiP)
DarkBit overview

DarkBit Module

MSSM/SingletDM

Process Catalog

Nucleon couplings

Gamma/Nu yields

Weff

Boltzmann solver

lnL indirect

lnL relic

lnL direct

Backends

DarkSUSY
- BRs, gamma-yields, relic density, Boltzmann solver

MicrOmegas
- relic density

NuLike
- IceCube

GamLike
- GC, dwarfs

DDcalc
- Xenon, LUX
C++ library with simple interface to most relevant likelihood functions from Fermi LAT and IACTs

Particle physics input:

\[
\frac{1}{m^2_{\chi}} \frac{d\sigma v}{dE}(v, E)
\]

Output: lnL

Uncertainties in the DM distribution (or astrophysical foregrounds) are internally marginalized over.

Correct treatment of energy dispersion and spectral singularities (lines, virtual internal Bremsstrahlung, boxes).