A FORTRAN PROGRAM FOR EXPERIMENTAL WIMP ANALYSIS

Based on the Mathematica package by Haxton-Berkeley group

Oliver Gorton

11 November 2020

In search of WIMPs (Weakly Interacting Massive Particles)

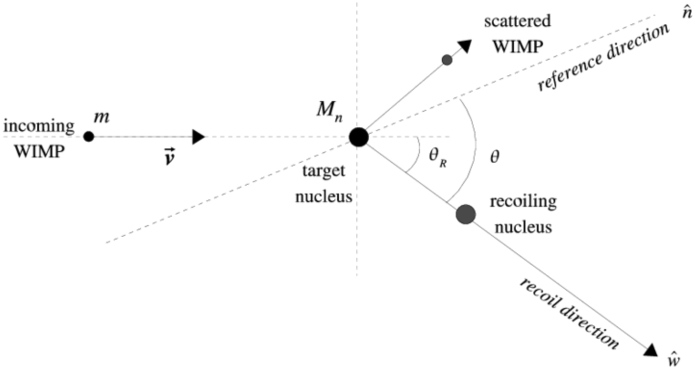


Fig. from [Alenazi and Gondolo, 2008]

Predict event rate

$$\frac{dR_D}{dE_R} = \frac{dR_D}{d\vec{q}^2}(q) = N_T n_\chi \int_{v_{min}}^{\infty} \frac{d\sigma(v,q)}{d\vec{q}^2} v \; \tilde{f}(\vec{v}) \; d^3 v \qquad (1)$$

q is the WIMP-nucleon momentum transfer
N_T is the number of target nuclei,
$n_\chi = \rho_\chi / m_\chi$ is the local dark matter density,
σ is the WIMP-nucleon cross section
$\tilde{f}(\vec{v})$ is dark matter velocity distribution in the lab-frame

Mathematica Package for (WIMP) Experimental Analysis

Model-independent WIMP Scattering Responses and Event Rates:
A Mathematica Package for Experimental Analysis

Nikhil Anand[1], A. Liam Fitzpatrick[2], W. C. Haxton[1]

August 30, 2013

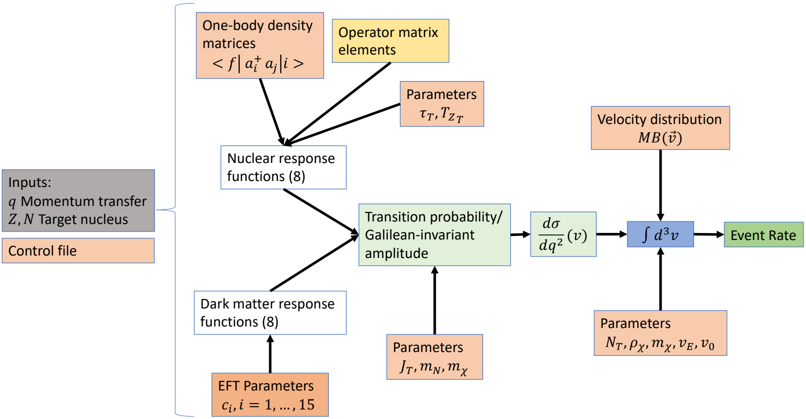
[1] *Dept. of Physics, University of California, Berkeley, 94720, and Lawrence Berkeley National Laboratory*
[2] *Stanford Institute for Theoretical Physics, Stanford University, Stanford, CA 94305*

References [Anand et al., 2014] [Fitzpatrick et al., 2013].

Reasons to write a FORTRAN version:

1. Speed
2. Interfacing with a parameter estimate workflow (e.g. MCMC)
3. Portable, *readable*, expandable, free

Code flow structure



Assume normal WIMP wind... boosted of course

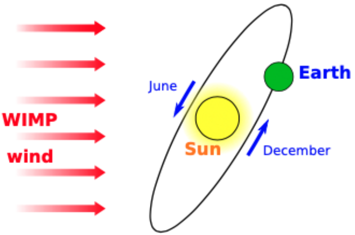
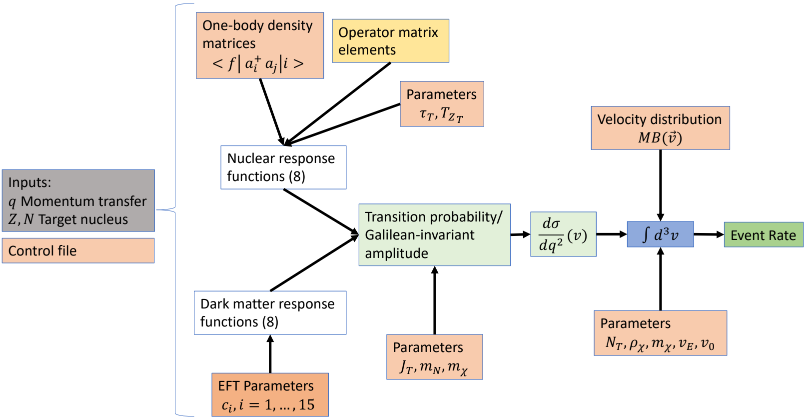


Fig. from [Freese et al., 2013]

$$\tilde{f}(\vec{v})d^3v = f(\vec{v} + \vec{v}_{earth})d^3v, \tag{2}$$

$$f(\vec{v})d^3v \propto e^{-\vec{v}^2/v_0^2}d^3v \tag{3}$$

Code flow structure



The cross section is determined from the Galilean invariant amplitude from EFT

$$\frac{d\sigma(v,q)}{dq^2} = 2m_T \frac{1}{4\pi v^2} T(v,q), \tag{4}$$

$$T(v,q) = \frac{4\pi}{2j_T+1} \frac{1}{(4m_\chi)^2} \sum_{\tau=0}^{1} \sum_{\tau'=0}^{1} \sum_{i=1}^{8} R_i^{\tau\tau'}(v^2,q^2) W_i^{\tau\tau'}(q) \tag{5}$$

- τ is isospin
- j_T target nucleus spin
- 8 terms: products of WIMP response functions R and nuclear response functions W
- Why 8? Starting from 15 general operators in LO, NLO, N^2LO in momenta and velocities, GIA expanded in spherical and vector spherical harmonics and averaged over spins, left with 8 combinations of 6 nuclear operators.

Code comparison

	Mathematica	FORTRAN
Files	1	25
Lines of code	1600*	4000
Subroutines	?	40 +
Si28 Eventrate runtime	1.8 s	0.12 s
Interface	Mathematica notebook	Interactive + control file

Mathematica: Sample code

dmformfactor-temp.m — v1.1 (git: master)

```
PhiPPsummand2[y_,{np_,lp_,jp_},{n_,l_,j_},jcap_]:=If[l==0,0,QNorm[l-1]*
Sqrt[l]*ThreeJ[{lp,0},{jcap+1,0},{l-1,0}]*BesselElementPlus[y,{np,lp},{n,l},jcap+1]*Sum[minus^(jcap-lcap)*JNorm[lcap]*SixJ[{jcap+1,1,lcap},{1,jcap,1}]*SixJ[{jcap+1,1,lcap},{l,lp,l-1}]*NineJSymbol[{lp,l,lcap},{1/2,1/2,1},{jp,j,jcap}],{lcap,jcap,jcap+1}]]
PhiPPsummand3[y_,{np_,lp_,jp_},{n_,l_,j_},jcap_]:=If[jcap==0,0,QNorm[l+1]*Sqrt[l+1]*ThreeJ[{lp,0},{jcap-1,0},{l+1,0}]*BesselElementMinus[y,{np,lp},{n,l},jcap-1]*Sum[minus^(jcap-lcap+1)*JNorm[lcap]*SixJ[{jcap-1,1,lcap},{1,jcap,1}]*SixJ[{jcap-1,1,lcap},{l,lp,l+1}]*NineJSymbol[{lp,l,lcap},{1/2,1/2,1},{jp,j,jcap}],{lcap,jcap-1,jcap}]]
PhiPPsummand4[y_,{np_,lp_,jp_},{n_,l_,j_},jcap_]:=If[jcap==0,0,If[l==0,0,QNorm[l-1]*
Sqrt[l]*ThreeJ[{lp,0},{jcap-1,0},{l-1,0}]*BesselElementPlus[y,{np,lp},{n,l},jcap-1]*Sum[minus^(jcap-lcap)*JNorm[lcap]*SixJ[{jcap-1,1,lcap},{1,jcap,1}]*SixJ[{jcap-1,1,lcap},{l,lp,l-1}]*NineJSymbol[{lp,l,lcap},{1/2,1/2,1},{jp,j,jcap}],{lcap,jcap-1,jcap}]]]
PhiPPJX[y_,{np_,lp_,jp_},{n_,l_,j_},jcap_]:=PhiPPoverall[y,{np,lp,jp},{n,l,j},jcap]*(QNorm[jcap+1]*Sqrt[jcap+1]*(PhiPPsummand1[y,{np,lp,jp},{n,l,j},jcap]+PhiPPsummand2[y,{np,lp,jp},{n,l,j},jcap])+If[jcap==0,0,QNorm[jcap-1]*Sqrt[jcap]*(PhiPPsummand3[y,{np,lp,jp},{n,l,j},jcap]+PhiPPsummand4[y,{np,lp,jp},{n,l,j},jcap])])
PhiPPJE[y_,{ncapp_,jp_},{ncap_,j_},jcap_]:=PhiPPJX[y,{Nodal[ncapp,jp],Lnumber[ncapp,jp],jp},{Nodal[ncap,j],Lnumber[ncap,j],j},jcap]
PhiPPJ[y_,{ncapp_,jp_},{ncap_,j_},jcap_]:=If[PhysicalConditionsNormal[ncapp,jp,ncap,j,jcap],PhiPPJE[y,{ncapp,jp},{ncap,j},jcap],0]

(**PhiTPJ**)
PhiPJX[y_,{np_,lp_,jp_},{n_,l_,j_},jcap_]:=PhiPPoverall[y,{np,lp,jp},{n,l,j},jcap]*(-QNorm[jcap+1]*Sqrt[jcap]*(PhiPPsummand1[y,{np,lp,jp},{n,l,j},jcap]+PhiPPsummand2[y,{np,lp,jp},{n,l,j},jcap])+If[jcap==0,0,QNorm[jcap-1]*Sqrt[jcap+1]*(PhiPPsummand3[y,{np,lp,jp},{n,l,j},jcap]+PhiPPsummand4[y,{np,lp,jp},{n,l,j},jcap])])
PhiPJE[y_,{ncapp_,jp_},{ncap_,j_},jcap_]:=PhiPJX[y,{Nodal[ncapp,jp],Lnumber[ncapp,jp],jp},{Nodal[ncap,j],Lnumber[ncap,j],j},jcap]
PhiPJ[y_,{ncapp_,jp_},{ncap_,j_},jcap_]:=If[PhysicalConditionsNormal[ncapp,jp,ncap,j,jcap],PhiPJE[y,{ncapp,jp},{ncap,j},jcap],0]
PhiTPJ[y_,{ncapp_,jp_},{ncap_,j_},jcap_]:=PhiPJ[y,{ncapp,jp},{ncap,j},jcap]+SigmaJ[y,{ncapp,jp},{ncap,j},jcap]/2


(***Operators of Abnormal Parity***)

(**DeltaJ**)
MJLDivQoverall[y_,{np_,lp_,jp_},{n_,l_,j_},jcap_,lcap_]:=minus^(lcap+j+1/2)*QNorm[lp]*QNorm[jp]*QNorm[j]*QNorm[jcap]*QNorm[lcap]*SixJ[{lp,jp,1/2},{j,l,jcap}]/Sqrt[4Pi]
MJLDivQsummand1[y_,{np_,lp_,jp_},{n_,l_,j_},jcap_,lcap_]:=-Sqrt[l+1] QNorm[l+1] SixJ[{lcap,1,jcap},{l,lp,l+1}] ThreeJ[{lp,0},{lcap,0},{l+1,0}] BesselElementMinus[y,{np,lp},{n,l},lcap]
MJLDivQsummand2[y_,{np_,lp_,jp_},{n_,l_,j_},jcap_,lcap_]:=Sqrt[l] QNorm[l-1] SixJ[{lcap,1,jcap},{l,lp,l-1}] ThreeJ[{lp,0},{lcap,0},{l-1,0}] BesselElementPlus[y,{np,lp},{n,l},lcap]
MJLDivQ[y_,{np_,lp_,jp_},{n_,l_,j_},jcap_,lcap_]:=MJLDivQoverall[y,{np,lp,jp},{n,l,j},jcap,lcap]*(MJLDivQsummand1[y,{np,lp,jp},{n,l,j},jcap,lcap]+MJLDivQsummand2[y,{np,lp,jp},{n,l,j},jcap,lcap])

DeltaPrime[y_,{np_,lp_,jp_},{n_,l_,j_},jcap_]:=-Sqrt[jcap]/QNorm[jcap]*MJLDivQ[y,{np,lp,jp},{n,l,j},jcap,jcap+1]+Sqrt[jcap+1]/QNorm[jcap]*MJLDivQ[y,{np,lp,jp},{n,l,j},jcap,jcap-1]
Deltaop[y_,{np_,lp_,jp_},{n_,l_,j_},jcap_]:=MJLDivQ[y,{np,lp,jp},{n,l,j},jcap,jcap]
DeltaJE[y_,{ncapp_,jp_},{ncap_,j_},jcap_]:=Deltaop[y,{Nodal[ncapp,jp],Lnumber[ncapp,jp],jp},{Nodal[ncap,j],Lnumber[ncap,j],j},jcap]
DeltaJ[y_,{ncapp_,jp_},{ncap_,j_},jcap_]:=If[PhysicalConditionsAbnormal[ncapp,jp,ncap,j,jcap],DeltaJE[y,{ncapp,jp},{ncap,j},jcap],0]

(**SigmaPJ**)
SigmaPrime[y_,{np_,lp_,jp_},{n_,l_,j_},jcap_]:=-Sqrt[jcap]/QNorm[jcap]*MJLSigma[y,{np,lp,jp},{n,l,j},jcap,jcap+1]+Sqrt[jcap+1]/QNorm[jcap]*MJLSigma[y,{np,lp,jp},{n,l,j},jcap,jcap-1]
SigmaPrimeJE[y_,{ncapp_,jp_},{ncap_,j_},jcap_]:=SigmaPrime[y,{Nodal[ncapp,jp],Lnumber[ncapp,jp],jp},{Nodal[ncap,j],Lnumber[ncap,j],j},jcap]
SigmaPJ[y_,{ncapp_,jp_},{ncap_,j_},jcap_]:=If[PhysicalConditionsAbnormal[ncapp,jp,ncap,j,jcap],SigmaPrimeJE[y,{ncapp,jp},{ncap,j},jcap],0]

(**SigmaPPJ**)
SigmaSecond[y_,{np_,lp_,jp_},{n_,l_,j_},jcap_]:=(Sqrt[jcap+1] MJLSigma[y,{np,lp,jp},{n,l,j},jcap,jcap+1]+Sqrt[jcap]
```

Line: 1 Mathematica Tab Size: 4

Fortran: Sample code

lib — vim dmresponse.f90 — 155×53

```fortran
!----------------------------------------------------------------------------
function dmrMJ(tau1, tau2, q, v, jchi)
    ! O. Gorton 2020.01.
    ! import modules from modules.f90
    use masses

    implicit none

    ! outputs
    REAL(kind=8) :: dmrMJ

    ! inputs
    integer, INTENT(IN) :: tau1, tau2
    REAL(kind=8), INTENT(IN) :: q
    REAL(kind=8), INTENT(IN) :: v
    REAL(kind=8), INTENT(IN) :: jchi

    ! functions called
    !REAL(kind=8) :: Cl

    dmrMJ = 0.25*Cl(jchi) * ( &
            (cvec(tau1)%c(5)*cvec(tau2)%c(5)*q*q + cvec(tau1)%c(8)*cvec(tau2)%c(8)) &
            * (v*v - q*q/(4*muT*muT)) &
            + cvec(tau1)%c(11)*cvec(tau2)%c(11)*q*q &
        ) + (cvec(tau1)%c(1) + cvec(tau1)%c(2) * (v*v - q*q/(4*muT*muT))) * ( &
            cvec(tau2)%c(1) + cvec(tau2)%c(2) * (v*v - q*q/(4*muT*muT)) &
        )

end function dmrMJ

!----------------------------------------------------------------------------
function dmrPhiPPJ(tau1, tau2, q, v, jchi)
    ! O. Gorton 2020.01.

    ! import modules from modules.f90
    use masses

    implicit none

    ! outputs
    REAL(kind=8) :: dmrPhiPPJ

    ! inputs
    REAL(kind=8), INTENT(IN) :: q
    REAL(kind=8), INTENT(IN) :: v
    REAL(kind=8), INTENT(IN) :: jchi
    integer, INTENT(IN) :: tau1, tau2

    ! functions called
    !REAL(kind=8) :: Cl

    dmrPhiPPJ = q*q/(16*mN*mN) * Cl(jchi) * (cvec(tau1)%c(12) &
```

87,1 14%

Fortran example input Si28

Annotated input file

```
1       ! Compute differential scattering rate
3.0     ! q : 3-momentum of scattering event
14      ! N neutrons
14      ! Z protons
si28    ! Control filename (Req. EFT coefficients)
sd      ! Single particle space file (BIGSTICK .sps)
si28w   ! One-body density file (BIGSTICK .res)
y       ! Yes, fill core nucleons
```

Fortran example control file Si28

Control file "si28.control"

```
# Coefficient matrix (non-relativistic)
# Ommitted values are assumed to be 0.0.
# c_i^t
# i = 1,...,16
# t = 0 protons, 1 neutrons
#control name    t      i       c_i^t
coefnonrel      1      5       3.1
fakekeyword   3.141
vearth 232.0
dmdens 1.0
intpoints 500
```

Alenazi, M. S. and Gondolo, P. (2008).
Directional recoil rates for wimp direct detection.
Physical Review D, 77(4).

Anand, N., Fitzpatrick, A. L., and Haxton, W. C. (2014).
Weakly interacting massive particle-nucleus elastic scattering response.
Physical Review C, 89(6).

Fitzpatrick, A. L., Haxton, W., Katz, E., Lubbers, N., and Xu, Y. (2013).
The effective field theory of dark matter direct detection.
Journal of Cosmology and Astroparticle Physics, 2013(02):004–004.

Freese, K., Lisanti, M., and Savage, C. (2013).
Colloquium: Annual modulation of dark matter.
Rev. Mod. Phys., 85:1561–1581.