



# Solar reflection of sub-GeV DM

- I. Direct detection of (sub-GeV) dark matter
- II. Dark matter in the Sun
- III. Monte Carlo simulation of solar reflection
- IV. Results
- V. Summary & Outlook

I.  
**Direct Detection of  
(sub-GeV) Dark Matter**

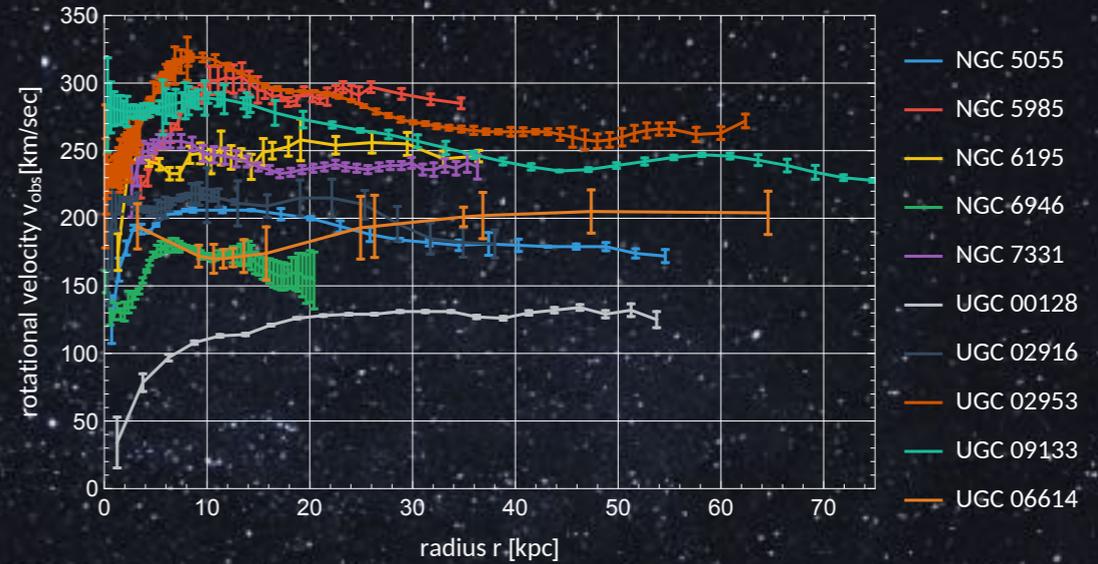
# How do we know about dark matter?

## (a) Fritz Zwicky and the Coma cluster



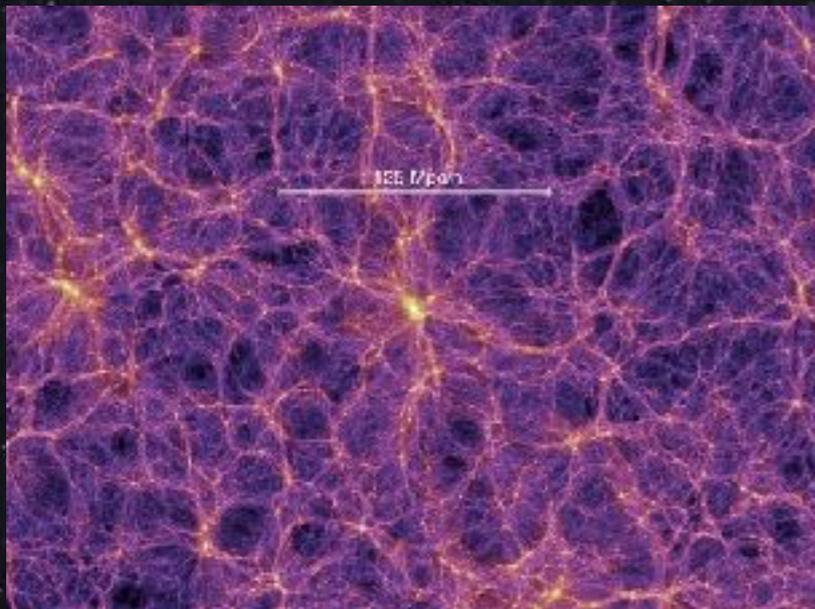
Photo: F. Clark (1971)  
F. Zwicky, *Helv. Phys. Acta* 6 (1933), 249

## (b) Galactic rotation curves



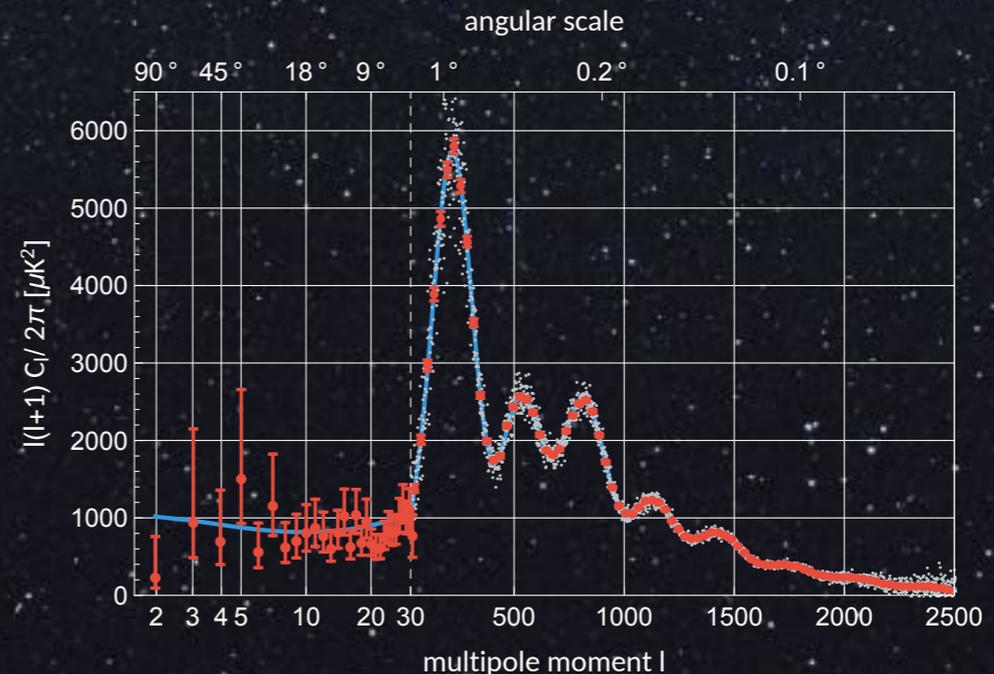
F. Lelli et al., *Astron. J.* 152 (2016), 157

## (c) Cosmological structure formation



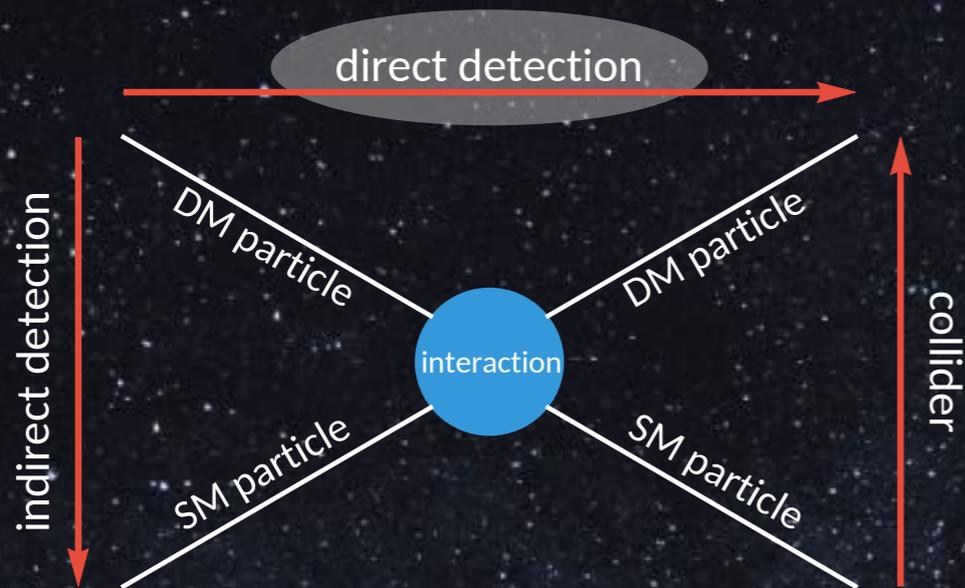
Springel et al., *Nature* 435 (2005), 629-636

## (d) Cosmic microwave background

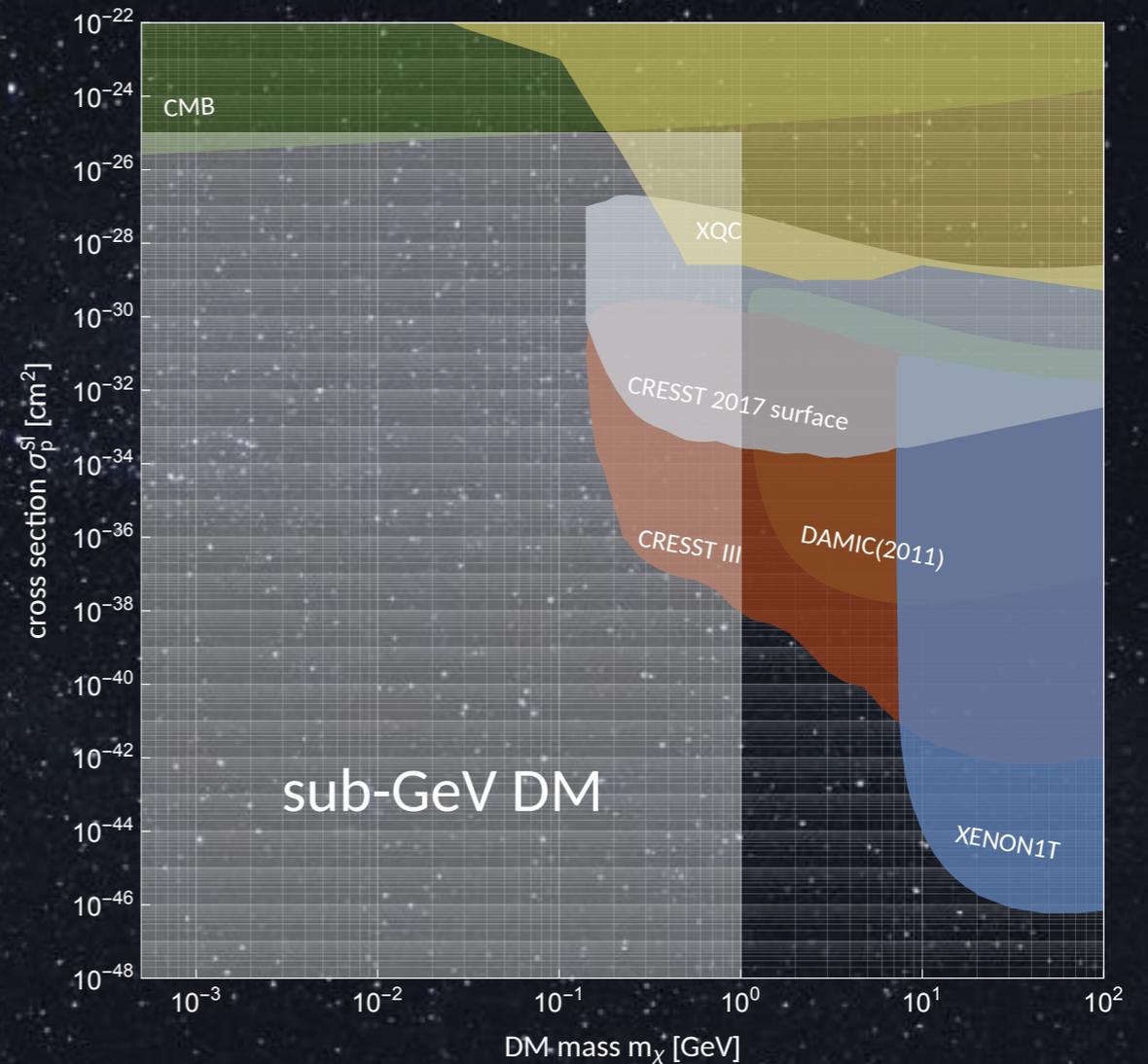


Planck Collaboration, *Planck Legacy Archive*, (2018)

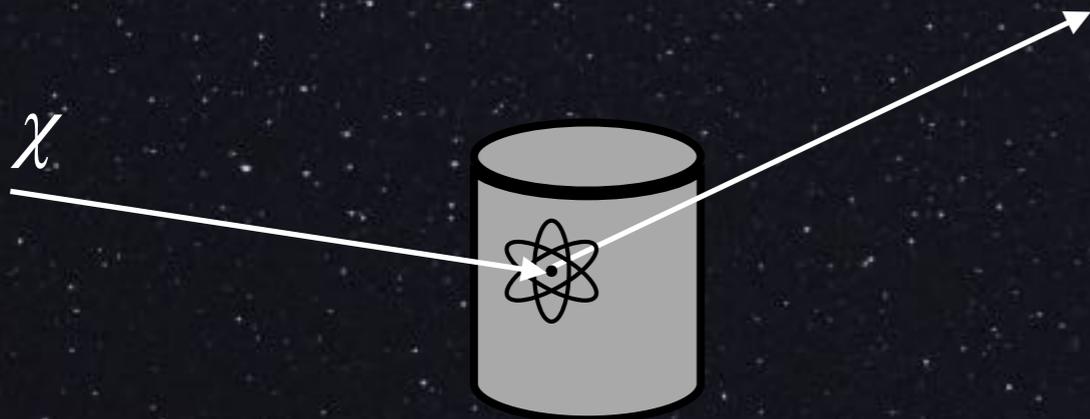
# Direct detection of dark matter



**Basic idea:** Measure the energy deposit of a DM-matter collision inside a detector.



- Conventional direct DM searches lose sensitivity to DM masses below  $\sim 1$  GeV.
- Why? And what can be done to probe lighter masses?



# The low mass frontier of DM searches

- Typical nuclear recoil:

$$E_R^{\text{typ}} \sim \frac{2m_\chi^2 v_\chi^2}{m_T} \approx 2 \text{ keV}$$

$$\times \left( \frac{m_\chi}{10 \text{ GeV}} \right)^2 \left( \frac{v_\chi}{10^{-3}} \right)^2 \left( \frac{m_T}{m_{\text{Xe}}} \right)^{-1}$$

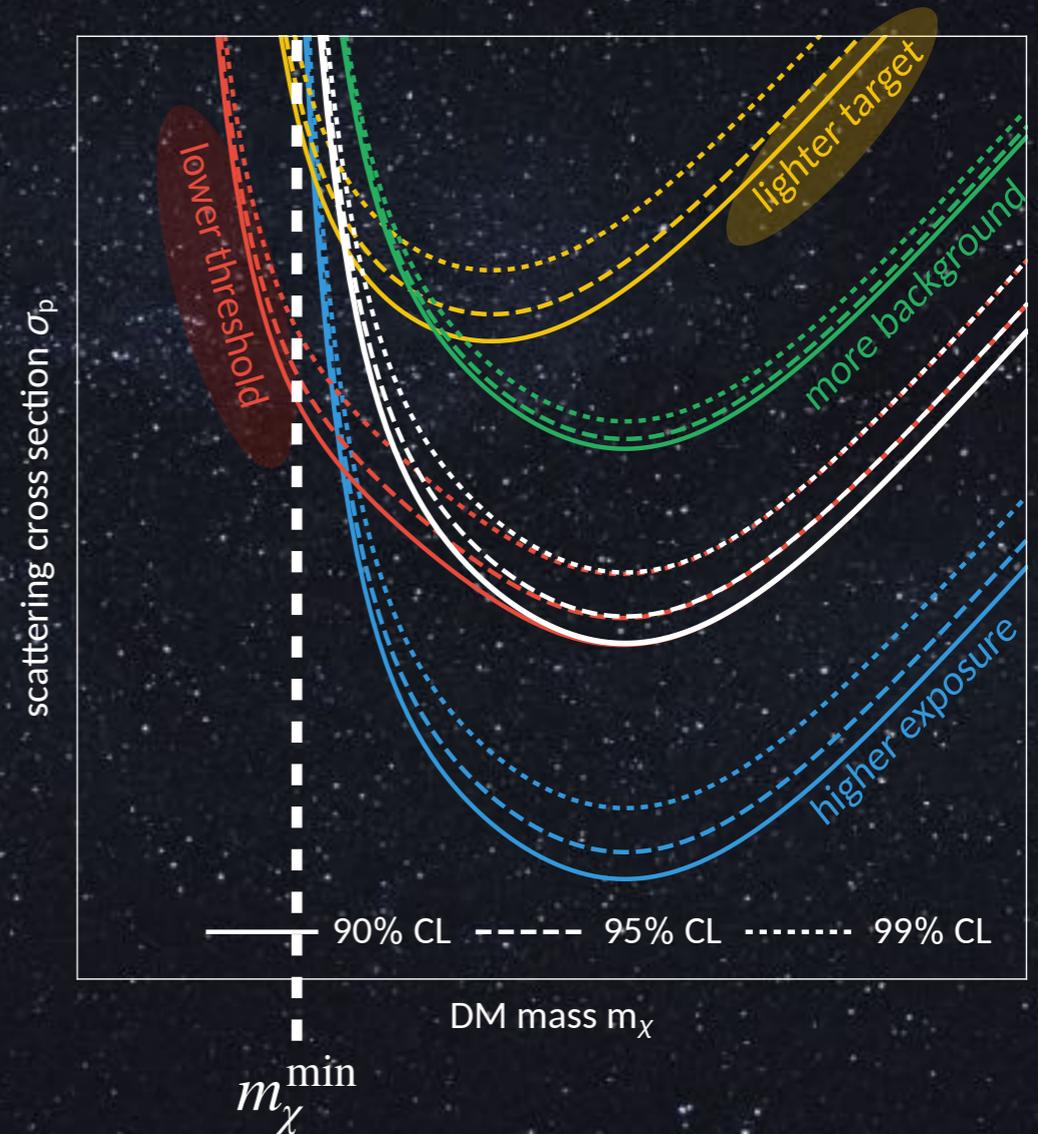
- Kinematic lower bound on the testable mass for a given recoil threshold.

$$m_\chi^{\text{min}} = \frac{m_T}{\sqrt{\frac{2m_T}{E_{\text{thr}}} v_{\text{max}} - 1}}$$

- Example: Threshold of XENON1T

$$E_{\text{thr}} \approx 5 \text{ keV} \Rightarrow m_\chi^{\text{min}} \approx 7 \text{ GeV}$$

Aprile et al., PRL 119 (2017)181301



Larger exposures do not probe lower masses.

# An incomplete list of search strategies for sub-GeV DM

- Bremsstrahlung emission from the recoiling nucleus.

Kouvaris & Pradler, PRL 118 (2017) 031803  
McCabe, PRD 96 (2017) 043010

- Migdal effect

Ibe et al., JHEP 1803 (2018) 194  
Dolan et al., PRL 121 (2018) 101801

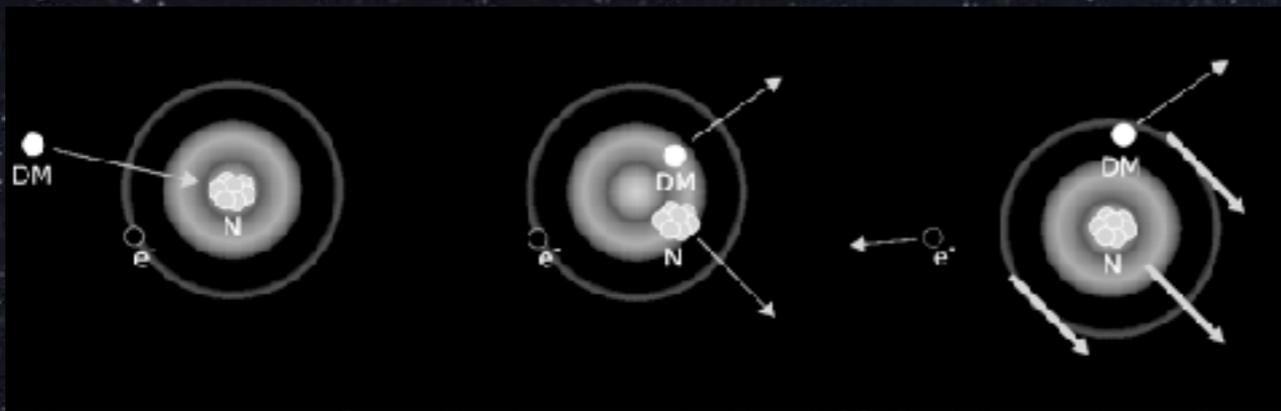


Figure from [1711.09906].

- Dirac materials (e.g. graphene) for directional detection.

Hochberg et al., Phys. Lett. B772 (2017) 239  
Coskuner et al., PRD, [arXiv:1909.09170]  
Geilhufe et al., PRD, [arXiv:1910.02091]

- Excitations of molecules in gases.

Essig et al., PRR, [arXiv:1907.07682]

- Up-scattering of DM by cosmic rays.

Bringmann & Pospelov, PRL 122 (2019) 171801  
Ema et al., PRL 122 (2019) 181802  
Alvey et al., PRL, [arXiv:1905.05776]  
Cappiello & Beacom, PRD, [arXiv:1906.11283]

# Direct Detection of Dark Matter via electron recoils

J. Kopp et al., Phys. Rev. D80 (2009) 083502

R. Essig et al., Phys. Rev. D85 (2012) 076007

Instead of nuclear recoils, search for DM-electron interactions.

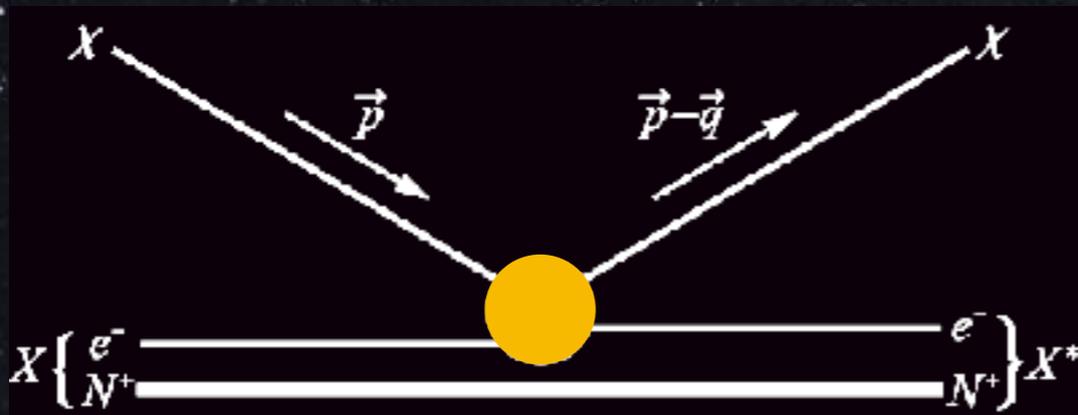


Figure from Essig et al., [arXiv:1509.01598]

- Different kinematics:

$$E_e^{\max} = \frac{1}{2} \mu_{\chi N} v^2 \lesssim E_\chi \sim 0.5 \text{ eV} \left( \frac{m_\chi}{\text{MeV}} \right)$$

Compare to

$$E_{\text{NR}}^{\max} = \gamma E_\chi$$

$$\gamma \approx 4 \frac{m_\chi}{m_N} \ll 1 \quad \text{for } m_\chi \ll m_N$$

- No kinematic penalty:  $E_e \leq E_\chi$ .
- The minimal test-able DM mass is

$$m_\chi \gtrsim \frac{2E_{\text{gap}}}{v_{\text{max}}^2}$$



# Solar Reflection of DM (SRDM)

H. An, M. Pospelov, J. Pradler, A. Ritz, PRL 120 (2018), 141801  
 TE, C. Kouvaris, N. G. Nielsen, PRD, 97 (2018), 063007

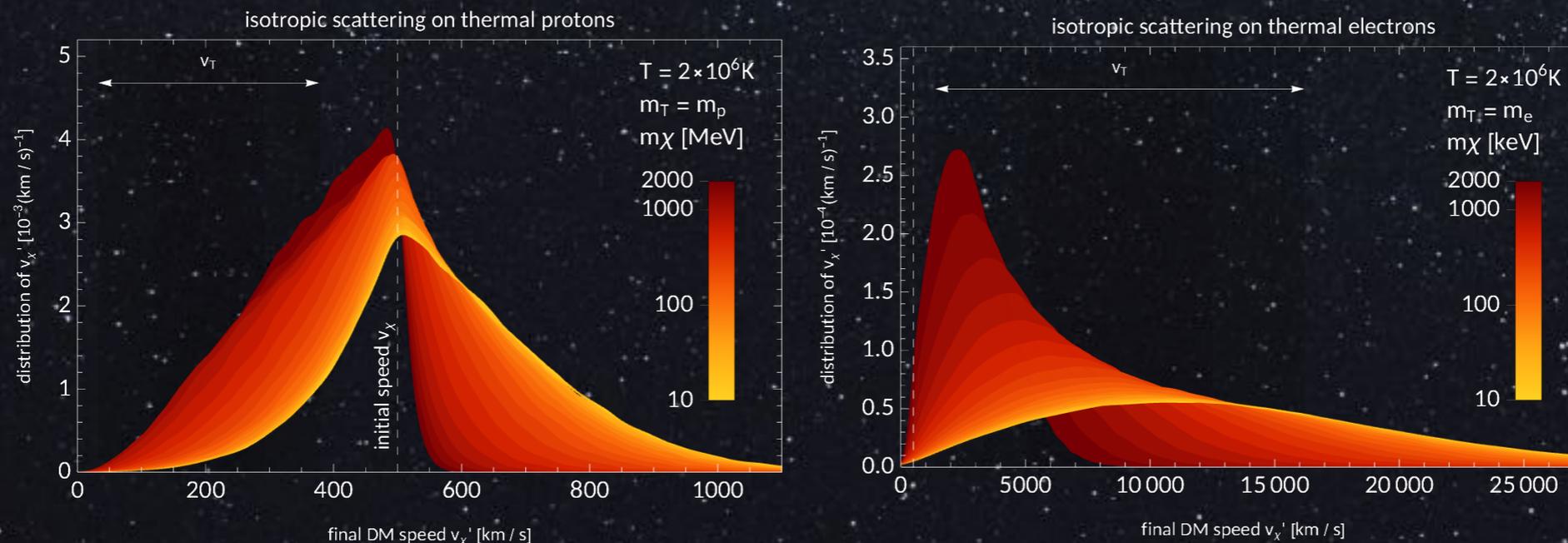
- Pro memoriam:

$$m_\chi \gtrsim \frac{m_T}{\sqrt{\frac{2m_T}{E_{\text{thr}}} v_{\text{max}} - 1}} \quad m_\chi \gtrsim \frac{2E_{\text{gap}}}{v_{\text{max}}^2}$$

- What if some process could increase the maximum DM speed somehow?

$$v_{\text{max}} > v_{\text{esc}}^{(\text{gal})} + v_{\oplus}$$

- Elastic scatterings on thermal targets inside the Sun could do just that.



$t = 0. \text{ hr}$

$N_s = 0$

$$v_\chi = 368 \frac{\text{km}}{\text{s}}$$

- $E_\chi/E_0 = 1.$

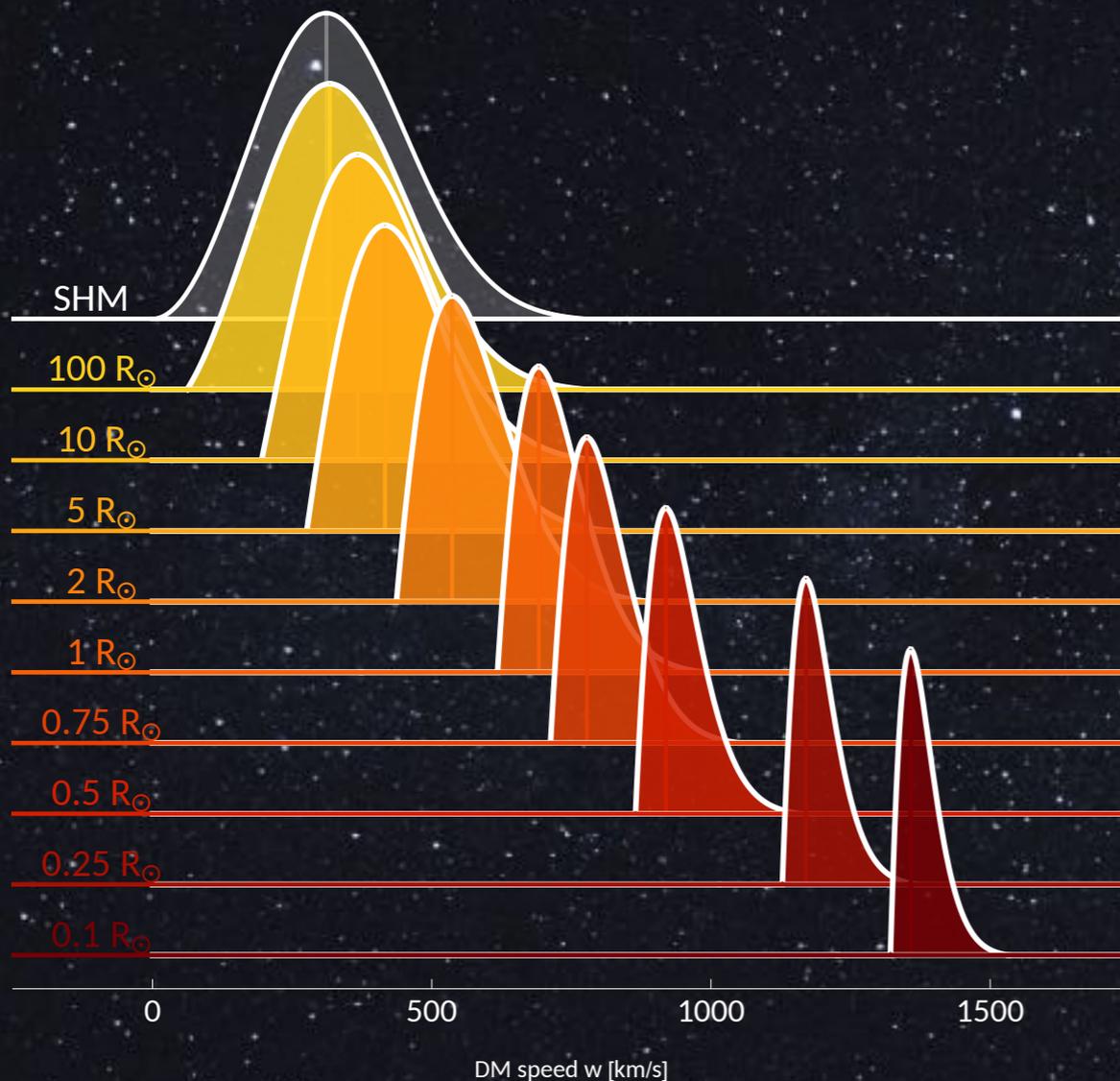


$m_\chi = 100 \text{ MeV}$

$\sigma_p = 0.2 \text{ pb}$

# DM particles passing through the Sun

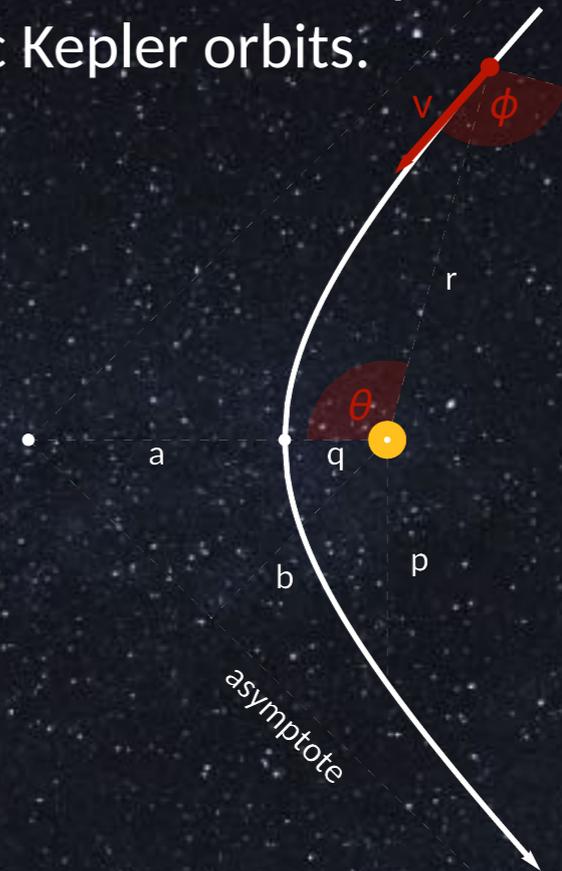
- Gravitational acceleration



$$w(u, r) = \sqrt{u^2 + v_{\text{esc}}(r)^2}$$

$$v_{\text{esc}}(r) = \sqrt{\frac{2G_N M_{\odot}}{r}}$$

- Outside the Sun, the DM particles follow hyperbolic Kepler orbits.



- The rate of DM particles entering the Sun is given by

$$\Gamma(m_{\chi}) = n_{\chi} \pi R_{\odot}^2 \int du f_{\text{halo}}(u) \left( u + \frac{v_{\text{esc}}(R_{\odot})^2}{u} \right)$$

$$\approx 1.1 \cdot 10^{33} \left( \frac{m_{\chi}}{\text{MeV}} \right)^{-1} \text{ s}^{-1}$$

# DM interaction models with heavy mediators

- We focused on the 4 standard scenarios in this study.

## 1. Spin-Independent nuclear interactions (SI)

$$\frac{d\sigma_N^{\text{SI}}}{dE_R} = \frac{m_N \sigma_p^{\text{SI}}}{2\mu_{\chi p}^2 v_\chi^2} \left[ Z + \frac{f_n}{f_p}(A - Z) \right]^2 \left| F_N(q) \right|^2$$

## 2. Spin-Dependent nuclear interactions (SD)

$$\frac{d\sigma_N^{\text{SD}}}{dE_R} = \frac{2m_N \sigma_p^{\text{SD}}}{3\mu_{\chi p}^2 v_\chi^2} \frac{J+1}{J} \left[ \langle S_p \rangle + \frac{f_n}{f_p} \langle S_n \rangle \right]^2 \left| F_N^{\text{SD}}(q) \right|^2$$

## 3. Electron interactions only.

$$\frac{d\sigma_e}{dq^2} = \frac{\bar{\sigma}_e}{4\mu_{\chi e}^2 v_\chi^2} F_{\text{DM}}(q)^2$$

## 4. Dark photon model

A small dark sector with an additional U(1) gauge group (broken) and kinetic mixing term.

$$\mathcal{L}_D = \bar{\chi}(i\gamma^\mu D_\mu - m_\chi)\chi + \frac{1}{4}F'_{\mu\nu}F'^{\mu\nu} + m_{A'}^2 A'_\mu A'^\mu + \frac{\varepsilon}{2}F_{\mu\nu}F'^{\mu\nu}$$

$$\frac{\bar{\sigma}_p}{\bar{\sigma}_e} = \left( \frac{\mu_{\chi p}}{\mu_{\chi e}} \right)^2$$

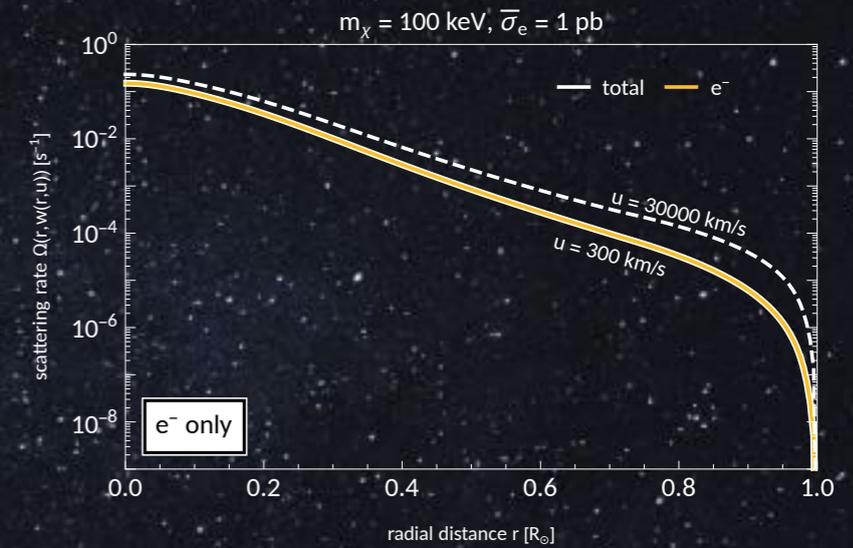
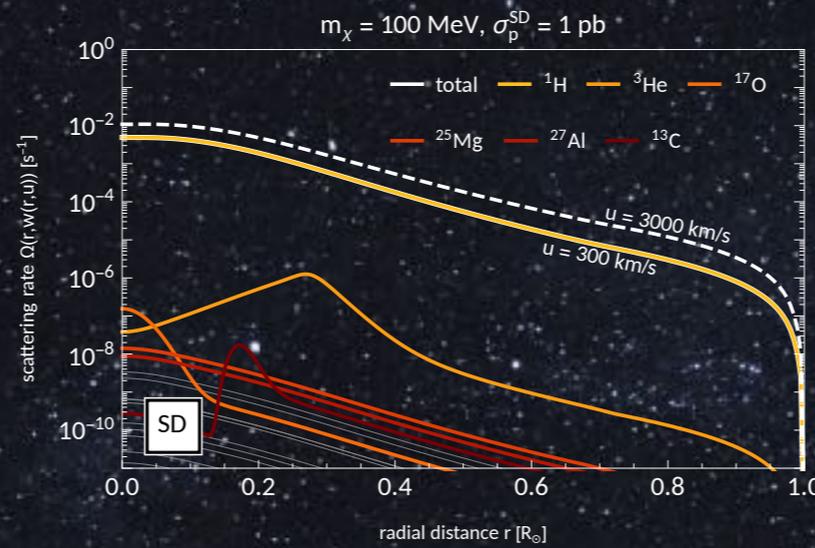
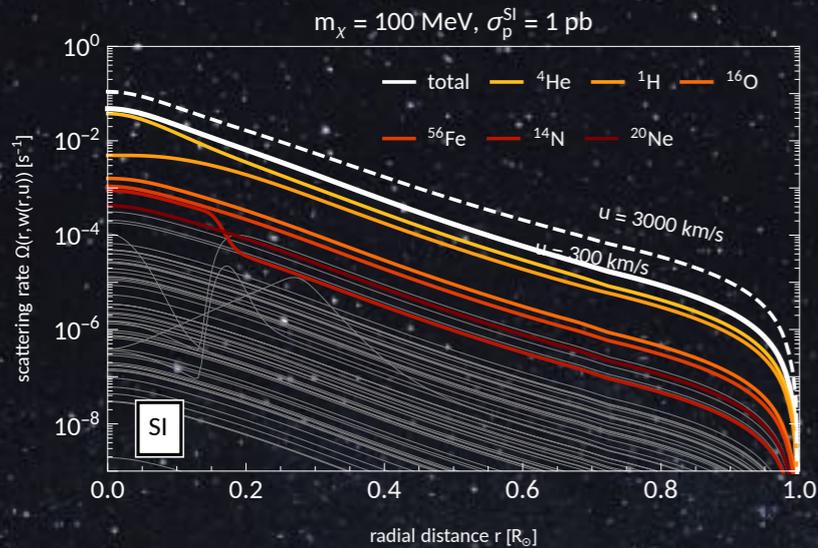
➔ Hierarchy of cross sections

# Underground DM scattering rates

$$\Omega(r, \mathbf{v}_\chi) = \sum_i n_i(r) \langle \sigma_i | \mathbf{v}_\chi - \mathbf{v}_{T,i} | \rangle$$

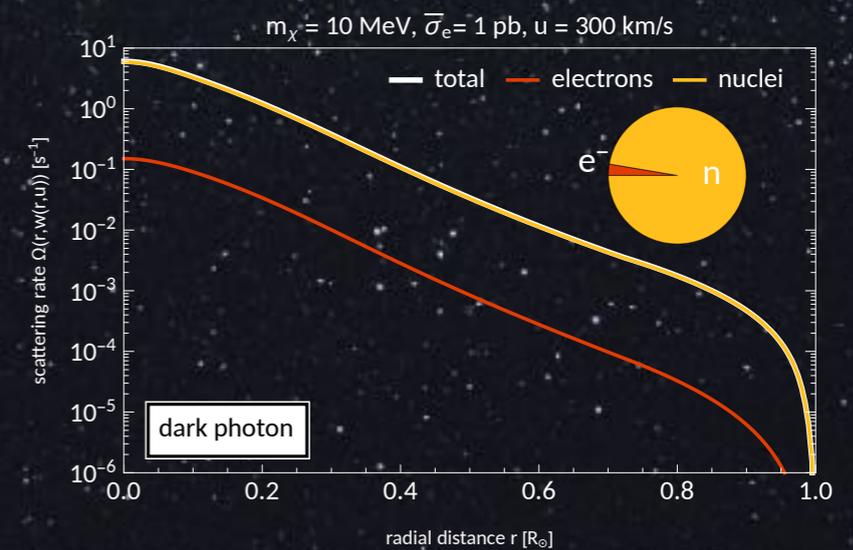
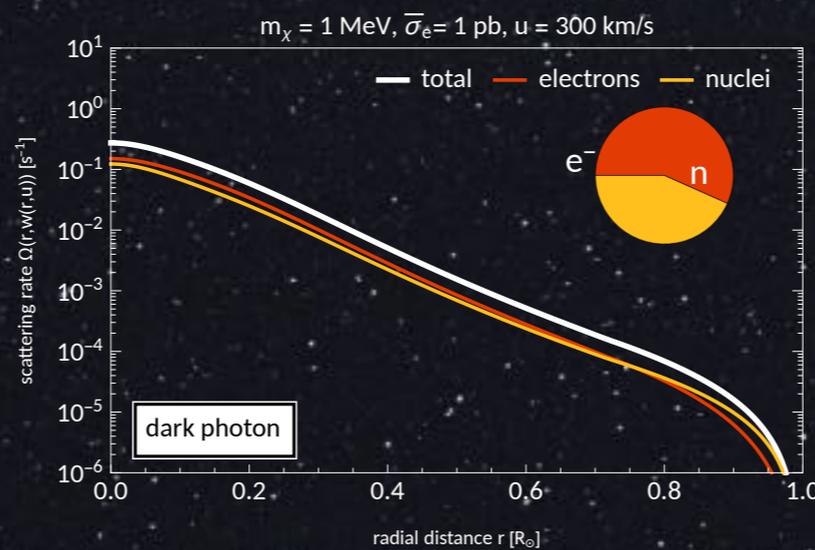
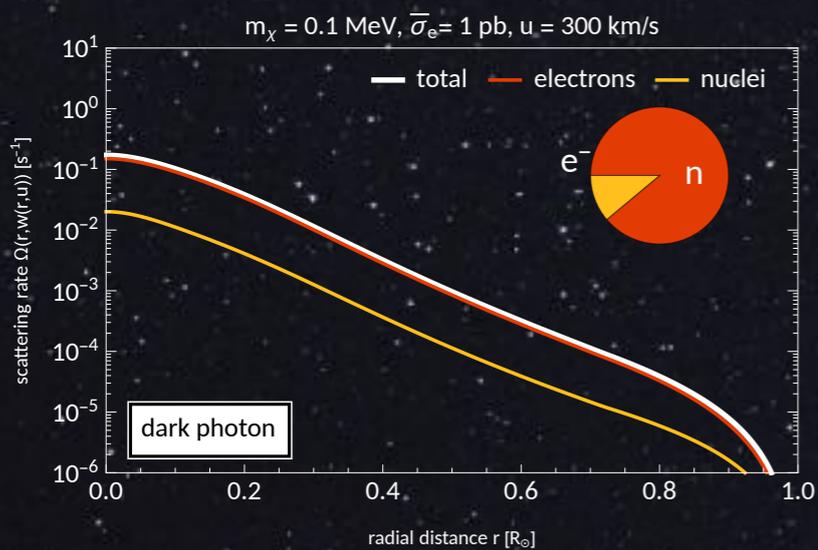
Number density  
Total cross section  
Relative speed

$\langle \cdot \rangle$  : Thermal average



- Dark photon model:

$$\frac{\Omega_e}{\Omega_p} \propto \frac{\bar{\sigma}_e \langle |\vec{v}_\chi - \vec{v}_e| \rangle}{\bar{\sigma}_p \langle |\vec{v}_\chi - \vec{v}_p| \rangle} = \left( \frac{\mu_{\chi p}}{\mu_{\chi e}} \right)^2 \times \mathcal{O}(10)$$



# First estimate of the SRDM flux

- We can make a first estimate of the total solar reflection rate.

$$\mathcal{R}_{\odot}(m_{\chi}) = p_{\text{refl}} \times \Gamma(m_{\chi})$$

- Here we assume an average probability for a DM particle to get reflected,  $p_{\text{refl}}$ .
- The total flux is therefore

$$\begin{aligned}\Phi_{\odot}(m_{\chi}) &= \frac{\mathcal{R}_{\odot}}{4\pi\ell^2} \\ &\approx 3.8 \cdot 10^5 p_{\text{refl}} \left( \frac{m_{\chi}}{\text{MeV}} \right)^{-1} \text{s}^{-1}\text{cm}^{-2}\end{aligned}$$

- Compare this to the halo DM flux

$$\Phi_{\text{halo}}(m_{\chi}) \approx 1.3 \cdot 10^{10} \left( \frac{m_{\chi}}{\text{MeV}} \right)^{-1} \text{s}^{-1}\text{cm}^{-2}$$

# Direct detection of SRDM

- The nuclear recoil spectrum caused by a generic DM flux is given by

$$\frac{dR}{dE_R} = N_T \int_{v_\chi > v_{\min}} dv_\chi \frac{d\Phi_\chi}{dv_\chi} \frac{d\sigma_{\chi N}}{dE_R}$$

- For halo DM, we usually directly insert

$$\frac{d\Phi_{\text{halo}}}{dv_\chi} = \frac{\rho_\chi}{m_\chi} v_\chi f_\chi(v_\chi)$$

- The SRDM flux can simply be added,

$$\frac{dR}{dE_R} = N_T \int dv_\chi \left( \frac{d\Phi_{\text{halo}}}{dv_\chi} + \frac{d\Phi_\odot}{dv_\chi} \right) \frac{d\sigma_{\chi N}}{dE_R}$$

Halo DM  
SRDM

- But one of the two fluxes will always dominate the other one.

# Distinguishing SRDM from background or halo DM

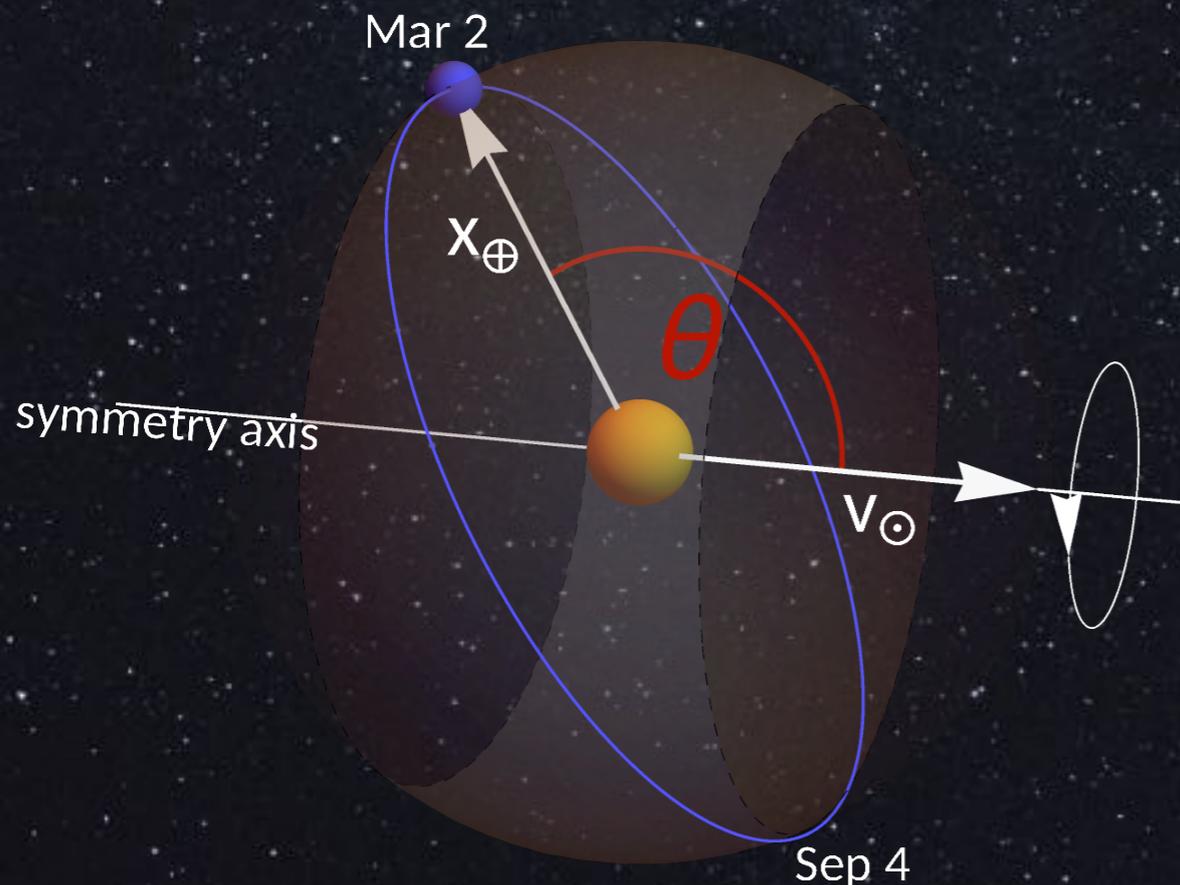
Signatures of a SRDM signal:

1. Directionality: If the detector is sensitive to the DM particle's incoming direction in the Sky, the Sun could be identified as the signal's origin.
2. Annual modulation: There are two source of annual modulation:
  - Orbital modulation: The flux scales as  $\ell^{-2}$ . Due to the Earth's orbital eccentricity this causes a small annual modulation.
  - Anisotropy modulation: If the Sun ejects the SRDM flux anisotropically, this could be a 2nd source of annual modulations.
3. Diurnal modulations: Underground scatterings inside the Earth would typically cause a day-night modulation.

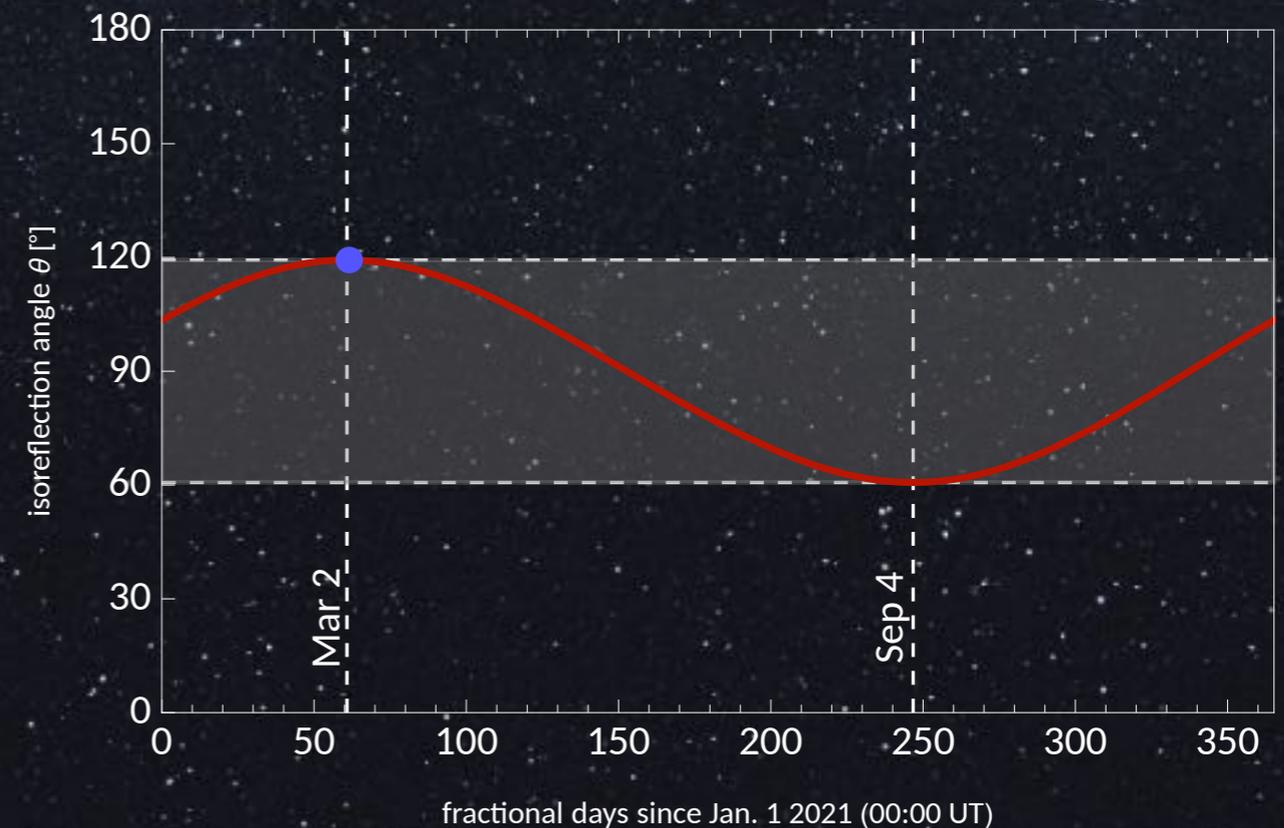
# The isoreflexion angle

- The boost into the Sun's rest frame gives rise to the DM wind.
- The system is symmetric under rotations around the Sun's velocity.
- We define the polar angle of this symmetry axis as the **isoreflexion angle**:

$$\theta = \angle(\mathbf{x}_{\oplus}, \mathbf{v}_{\odot})$$



- Time evolution of the Earth's isoreflexion angle



```
... //9. New velocity Eigen::Vector3d v2 = sqrt(G*Newton*Sun/p)*e*sin(theta2)*x2.normalized()
// double F2 = ncosh((e+cos(theta2))/(1.0+e*cos(theta2))) // double M2 = asinh(F2)-F2; // double t2 = sq
stance at which we sample from the halo distribution double R=100.0*AU; if(R<100.0*rSun) { cerr <<"Warning in
tachelers!!_1,vEarth); double u = Rejection_Sampling(pdf,0.0,(vesc+vEarth),ymax,PRNG); //Velocity Directio
dom Point on a flat disk at distance R Eigen::Vector3d az=-vini.normalized(); Eigen::Vector3d ox(0,az/2
cos(phi)+e*cos(phi)+ey); // cout <<"Norm = "<<(R*ex, norm); // cout <<"Norm = "<<(R*ex, norm); //
[0]/rSun<<"\t"<<IC.Position()[1]/rSun<<"\t"<<IC.Position()[2]/rSun<<"\t"<<IC.Radius()/rSun<<"\t"<<IC.Velocity(
r_Shift(IC,rMax,model); } //2. Orbit simulation //right hand sides of the 1st order equations of motion, double
Euler Crmer // void EC_Step(double dt,double &r,double &v,double &phi,double J,double dt,Sun::Model &model) // {
// //AK coefficients: // double k_r[4]; // double k_v[4]; // double k_p[4]; // k_r[0]= dt*dvdt(v); // k_v[0]
= dt*dvdt(r+k_r[1]/2.0,J,model); // k_p[2]= dt*dphidt(r+k_r[1]/2.0,J); // k_r[3]= dt*dvdt(v+k_v[2]
2)+k_v[3]); // phi+= 1.0/6.0*(k_p[3]+2.0*k_p[2]+k_p[1]); // //ODE integration with Runge Kutta Fehl
k_v[0]= dt*dvdt(r,J,model); k_p[0]= dt*dphidt(r,J); k_r[1]= dt*dvdt(v+k_v[0]/4.0); k_v[1]= dt*dvdt(r+k_r[0]
.0+k_r[1],J); k_r[3]= dt*dvdt(v+1932.0/2197.0*k_v[0]-7200.0/2197.0*k_v[1]+7296.0/2197.0*k_v[2]); k_v[3]= dt
.0*k_v[1]+3680.0/513.0*k_v[2]-845.0/4104.0*k_v[3]); k_v[4]= dt*dvdt(r+439.0/216.0*k_r[0]-8.0*k_r[1]
-3594.0/2809.0*k_r[2]+1859.0/4104.0*k_r[3]-11.0/40.0*k_r[4]); k_v[5]= dt*dvdt(r-8.0/27.0*k_r[0]-2.0*k_r[1]-3M
ble r4= r+28.0/216.0*k_r[0]+104.0/2809.0*k_r[2]+2107.0/4104.0*k_r[3]-1.0/8.0*k_r[4]; double v4= v+28.0/216.
k_r[2]+28561.0/56430.0*k_r[3]-9.0/50.0*k_r[4]+2.0/55.0*k_r[5]; double v5= v+16.0/135.0*k_v[0]+6656.0/12025.0*k
r4,abs(v5-v4),abs(phi5-phi4)); std::vector<double> tolerance = {1.0*km,1e-3*km/sec,1e-6}; //New steps:ie
s),std::end(deltas)); //Next steps if(err[0]<tolerance[0]&&err[1]<tolerance[1]&&err[2]<tolerance[2]) { if(r<
xi -= speed*dt/mfp; } t+= dt; r = r4; v = v4; phi = phi4; dt = std::min(dtmin, std::min(delta*dt,dtMax)); } else { dt
vector<Eigen::Vector3d> &axes) { double v_phi = J/pow(r,2); Eigen::Vector3d xNew = r*(cos(phi)*axes[0]+sin(phi)*ax
scattering or (b) leaving the sun. bool Propagate(Event& x0,DM_Particle& DM,Sun::Model& model,std::m
)).normalized(); Eigen::Vector3d ax_y = ax_z.cross(ax_x); std::vector<Eigen::Vector3d> axes = {ax_x,ax_y
(())}.dot(ax_z); //Sample -log(1-xi) double logXi = -1.0*log(1.0-ProbabilitySample(PRNG)); //Start integ
_step(t,r,v_r,phi,J,dt,logXi,DM,model); if(Mxi0==0) { Event xNew =PlaneTo3D(t,r,phi,v_r,J,axes); r <<xNew.Positi
eed(model.vEsc2(xNew.Radius())) <<"\t" <<InUnits(xNew.Speed(),km/sec) <<endl; } // double dd = (xOld.Position
num // t <<xNew.Position()[0]/rSun<<"\t"<<xNew.Position()[1]/rSun<<"\t"<<xNew.Position()[2]/rSun<<"\t" <<t/sec
Old<rMax&&rMax&&(v_r+v_r+J)/r)>model.vEsc2(r) { propagate=false; x0=PlaneTo3D(t,r,phi,v_r,J,axes); } else
(Event& x0,DM_Particle& DM,Sun::Model& model,std::mt19937& PRNG) { // double speed0=x0.Speed(); // cout <<"Scatt
d inates{1.0,ThetaSample(PRNG),PhiSample(PRNG)}; double vRel = (vTarget-x0.Velocity()).norm(); Eige
eed0<<endl; // if(speed0>sqrt(model.vEsc2(r))&&vnew.norm()<sqrt(model.vEsc2(r))) cout <<"\tcapture"<<endl; //
ol& model, unsigned int &nScattering, std::mt19937& PRNG) { //Save trajectory of stream
x.Speed()*x.Speed()-model.vEsc2(x.Radius())/E0<<endl; //Output // Event x = IC; bool success; //Counters nScat
if(r<rSun) { if(nScattering>nScattering_max) { success = false; break; } // double E1=DM.mass/2.0*(x
*(x.Speed()*x.Speed()-model.vEsc2(x.Radius())/E0<<endl; // if(E1>0 && E2<0) { // cout <<"capture"<<endl;
<"\t" <<r/rSun << endl; // // r.close(); // // // if(x.Speed()) <sqrt(model.v
bation()[0]/rSun<<"\t"<<xFinal.Position()[1]/rSun<<"\t"<<xFinal.Position()[2]/rSun<<"\t"<<xFinal.Radius
ces=false; if(x.time()/sec<9999) return false; return success; } Result Generate_Data(Configur
r,numprocs-1)? 0 : config.rank-1; int tag =0; MPI_Status status; MPI_Request req; //Datapoin
data; data.resize(config.nRings); //Counters //Total number of simulated particles unsigned long int G_Coun
vector<unsigned long int> L_Counter_Data_new(config.nRings,0); //Number of passes // std::vector<unsigned
ber of scatterings; std::vector<double> G_nScattering_Av(config.nRings,0.0); std::vector<double> L_nSca
Isend(&G_Counter_Data.front(),config.nRings,MPI_UNSIGNED_LONG,destination,tag,MPI_COMM_WORL
G)) unsigned int nScattering=0; bool success = Simulate_Trajectory(x,DM,model,nScattering,PRNG);
_nScattering_Av[Isorting]+nScattering)/L_Counter_Data[Isorting]; double speed=x.Speed(); r[nS
DHN_WORLD,&flag,&status); if(flag) { //Receive the tokens MPI_Recv(L_Counter_Data.front(),config.nRi
r_Data[1]=L_Counter_Data_new[1]; L_Counter_Data_new[1]=0; } // cout <<config.rank <<"\t" <<G_Coun
else if (nRings>config.nData) tag = status.MPI_TAG; //Pass on the tokens, unless you are the very la
gth cout <<"r"; for(int i=0;i<2.0*BarLength;i++)cout <<" "; cout <<"r"; for(int i=0;i<BarLength
) cout <<"r"; for(int j=0;j<10*BarLength;j++) cout <<" " cout <<"r"; } // for(int i=0
_Counter_Free,1,MPI_UNSIGNED_LONG_LONG,MPI_SUM,MPI_COMM_WORLD); MPI_Allreduce(&L_nSca
> Global_Data; for(int i = 0;i<config.nRings;i++) { unsigned long int Global_SampleSize; MPI
cy_displs[config.numprocs]; MPI_Allgather(&L_Counter_Data[i],MPI_UNSIGNED_LONG,&nData,Loc
SampleSize); // MPI_Gatherv(&data[i].front(),data.size(),mp_datapoint,&ring_data.front(
) MPI_Barrier(MPI_COMM_WORLD); // for(int i=0;i<config.nRings;i++) // { // if(config.n
h,1,MPI_DOUBLE,MPI_MAX,MPI_COMM_WORLD); return Result(Global_Data,DM,G_Counter_Simulati
std::vector<double> &nscav,double dt) { data=dat; particle = p; // sample_size
} nscattering_mean=nscav; nscattering_mean_tot=0.0; for(unsigned int i=0;i
n &config) { if(config.rank==0) { std::cout <<"Simulation summa
<Round(1.0*nSimulations/duration)<<"(sec)"<<std::endl; // (config.nRings==1) st
<" " <<Round(100.0*nFails/nSimulations)<<" " <<std::endl <<"\tDuration[s]:"
ize[i] <<"\t\t\t" <<Round(nScatterings_mean[i])<<std::endl; } } #inc
ructions.cpp" using namespace libconfig; //1. Struct with input param
nData=nD; //Parallelisation rank = myrank; numprocs=ap; } Configur
const fileIOException &fileex) { std::cerr <<"I/O error while re
try { size = cfg.lookup("simID").c_str(); } catch(const
file." << endl; exit(EXIT_FAILURE); } //Sers
"one."<<endl<<endl; } //2. Event
```

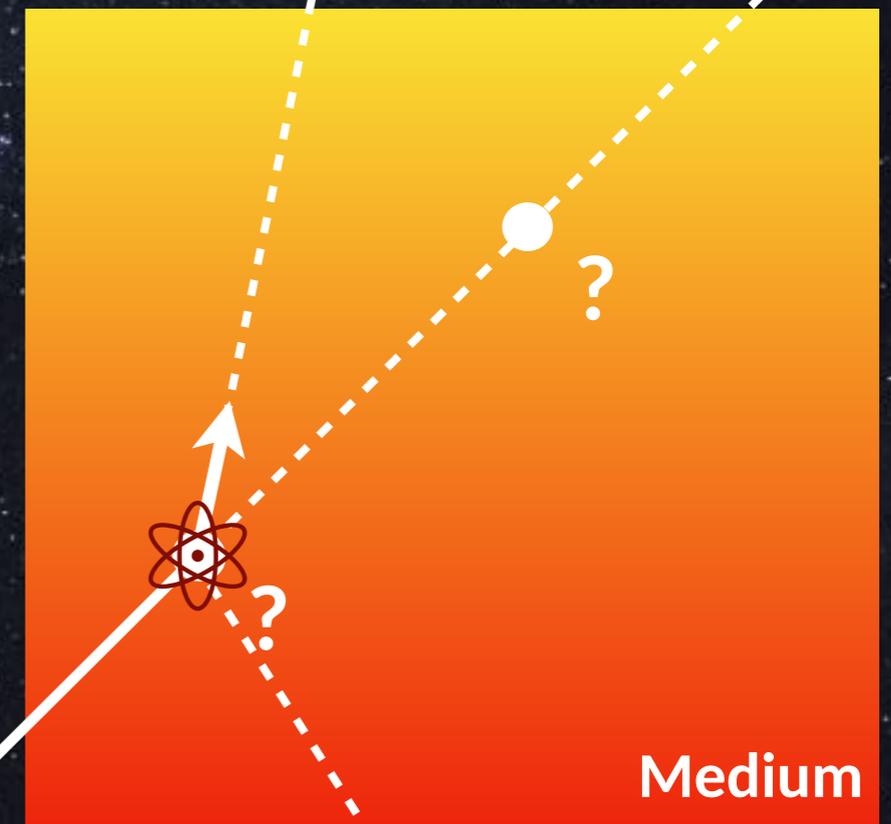
# III. Monte Carlo simulation of solar reflection

# MC Simulations of DM in the Sun

The main random processes are

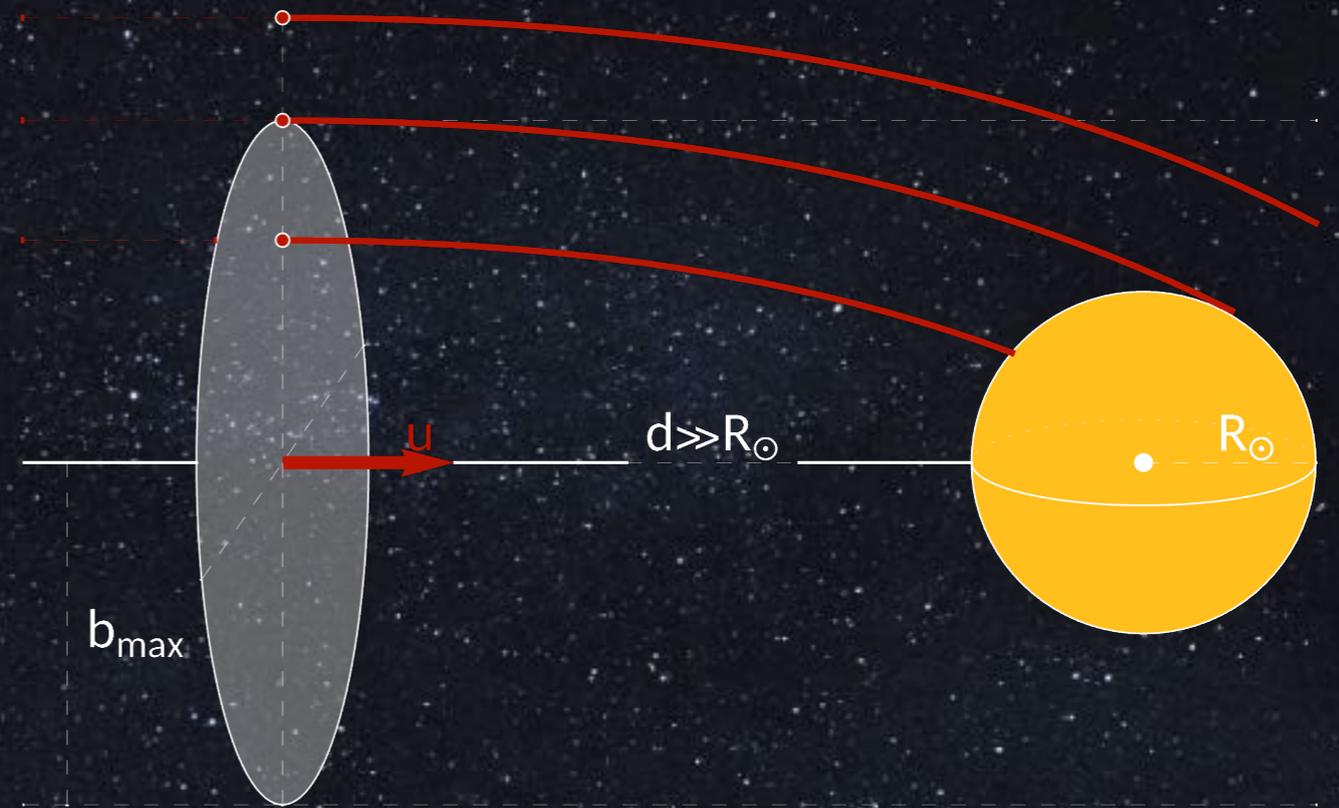
1. **Initial Conditions:** Where does the particle start?
2. **Free distance:** Where does the particle scatter?
3. **Target:** What does the particle scatter on? How fast is it?
4. **Scattering angle:** How does the particle scatter?

**Repeat steps 2-4.**



# Initial conditions

1. Sample a velocity  $u$  far away from the Sun from the halo.
2. Sample the initial position with a maximum impact parameter.
3. Propagate the particle analytically on its hyperbolic Kepler orbit to a location close to the Sun.



# Free propagation

- The probability for a DM particle to propagate underground without scattering:

$$P(\Delta t) = 1 - \exp\left(-\int_0^{\Delta t} \frac{dt}{\tau(r(t), v(t))}\right)$$

- Here, we used the mean free time

$$\tau(r, v) = \Omega(r, v)^{-1}$$

- This enables us to find the location of the next scattering.
- We solve the equations of motion iteratively using the Runge-Kutta-Fehlberg method. The particle scatters as soon as

$$\sum_i \frac{\Delta t_i}{\tau(r_i, v_i)} > -\ln(\xi)$$

# Scatterings inside the Sun

- Once we know that the DM particle scatters at a radial distance  $r$ , we can identify the target species. The probability to scatter on target  $i$  is

$$P_i = \frac{\Omega_i(r, v_\chi)}{\Omega(r, v_\chi)}$$

- The target's velocity can be sampled from the following distribution,

$$f(\mathbf{v}_T) = \frac{|\mathbf{v}_\chi - \mathbf{v}_T|}{\langle |\mathbf{v}_\chi - \mathbf{v}_T| \rangle} f_i(\mathbf{v}_T, T) \quad \text{Maxwell-Boltzmann distribution}$$

- Since we assume contact interactions with heavy mediators, the scattering is isotropic.
- Finally, the new DM velocity is given by

$$\mathbf{v}'_\chi = \frac{m_T |\mathbf{v}_\chi - \mathbf{v}_T|}{m_T + m_\chi} \mathbf{n} + \frac{m_\chi \mathbf{v}_\chi + m_T \mathbf{v}_T}{m_T + m_\chi}$$

$t = 0. \text{ hr}$

$N_s = 0$

$$v_\chi = 456 \frac{\text{km}}{\text{s}}$$

$$E_\chi/E_0 = 1.$$



$m_\chi = 100 \text{ MeV}$

$\sigma_p = 0.2 \text{ pb}$

Emken 2019

# From MC simulation to the SRDM flux

- The total reflection rate and flux is found given by the amount of reflections among the simulated particles.

$$\mathcal{R}_{\odot}^{\text{MC}} = \frac{N_{\text{refl}}}{N_{\text{sim}}} \Gamma(m_{\chi}) \quad \Phi_{\odot}^{\text{MC}} = \frac{\mathcal{R}_{\odot}^{\text{MC}}}{4\pi\ell^2} \quad \ell = 1 \text{ AU}$$

- The reflection spectrum is based on the speed data recorded at 1 AU distance. Instead of histograms, we prefer a smooth distribution and use Kernel density estimation (KDE).

$$\frac{d\mathcal{R}_{\odot}}{dv_{\chi}} = \mathcal{R}_{\odot}^{\text{MC}} f_{\odot}^{\text{KDE}}(v_{\chi}) \quad \frac{d\Phi_{\odot}^{\text{MC}}}{dv_{\chi}} = \frac{1}{4\pi\ell^2} \frac{d\mathcal{R}_{\odot}}{dv_{\chi}}$$

- To study anisotropies of the SRDM flux, we define equal-area isoreflexion rings and perform the analysis for each ring.



# The Simulation Code

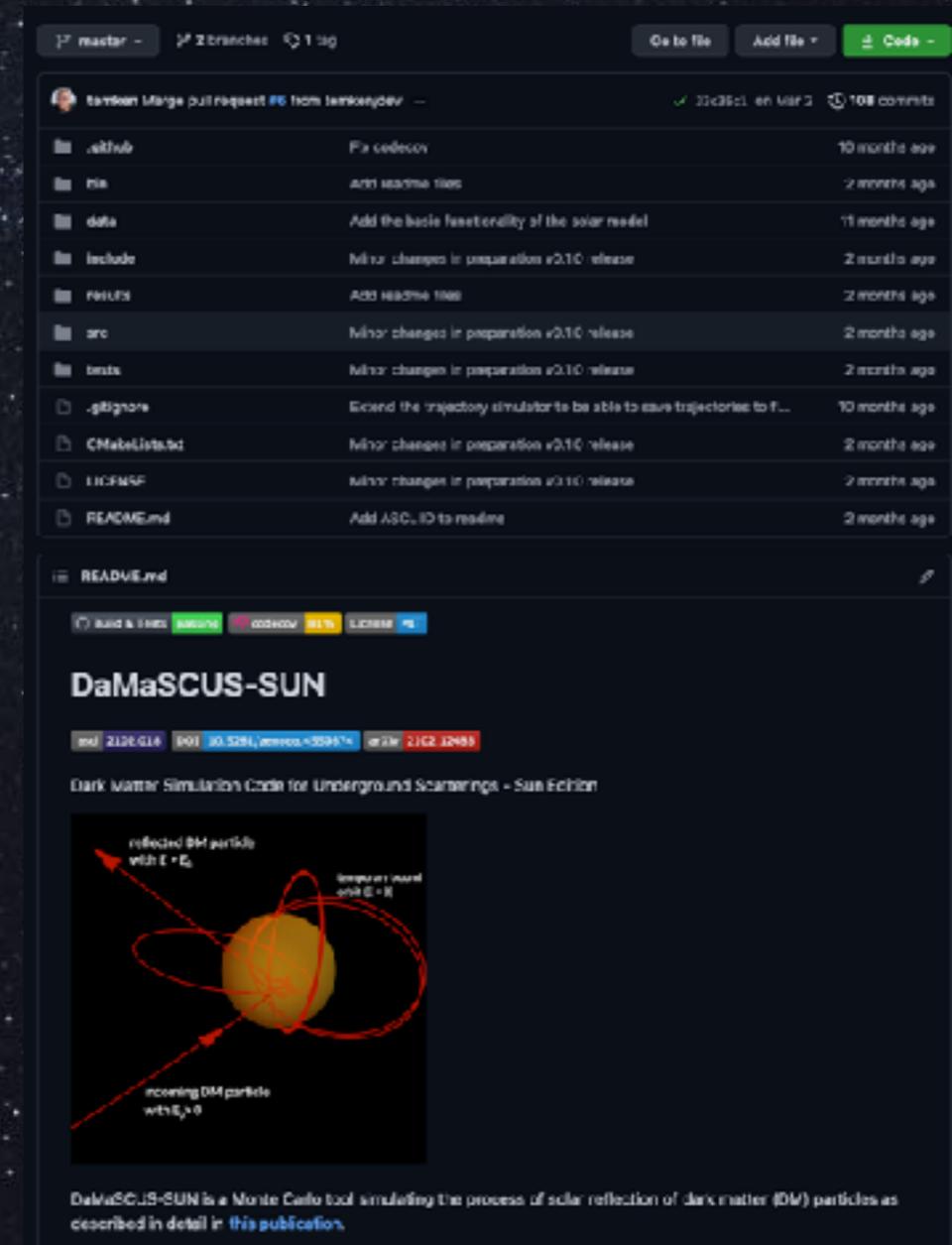
"The code is what you did, the paper is what you think you did."

-Patrick Koppenburg on Twitter

- The *Dark Matter Simulation Code for Underground Scatterings - Sun Edition* (DaMaSCUS-SUN) is publicly available on *github*.

<https://github.com/temken/DaMaSCUS-SUN>

- Written in C++, built with CMake, set up with unit and build testing, code coverage,...
- Fully parallelized (MPI), ready to run on HPC clusters.

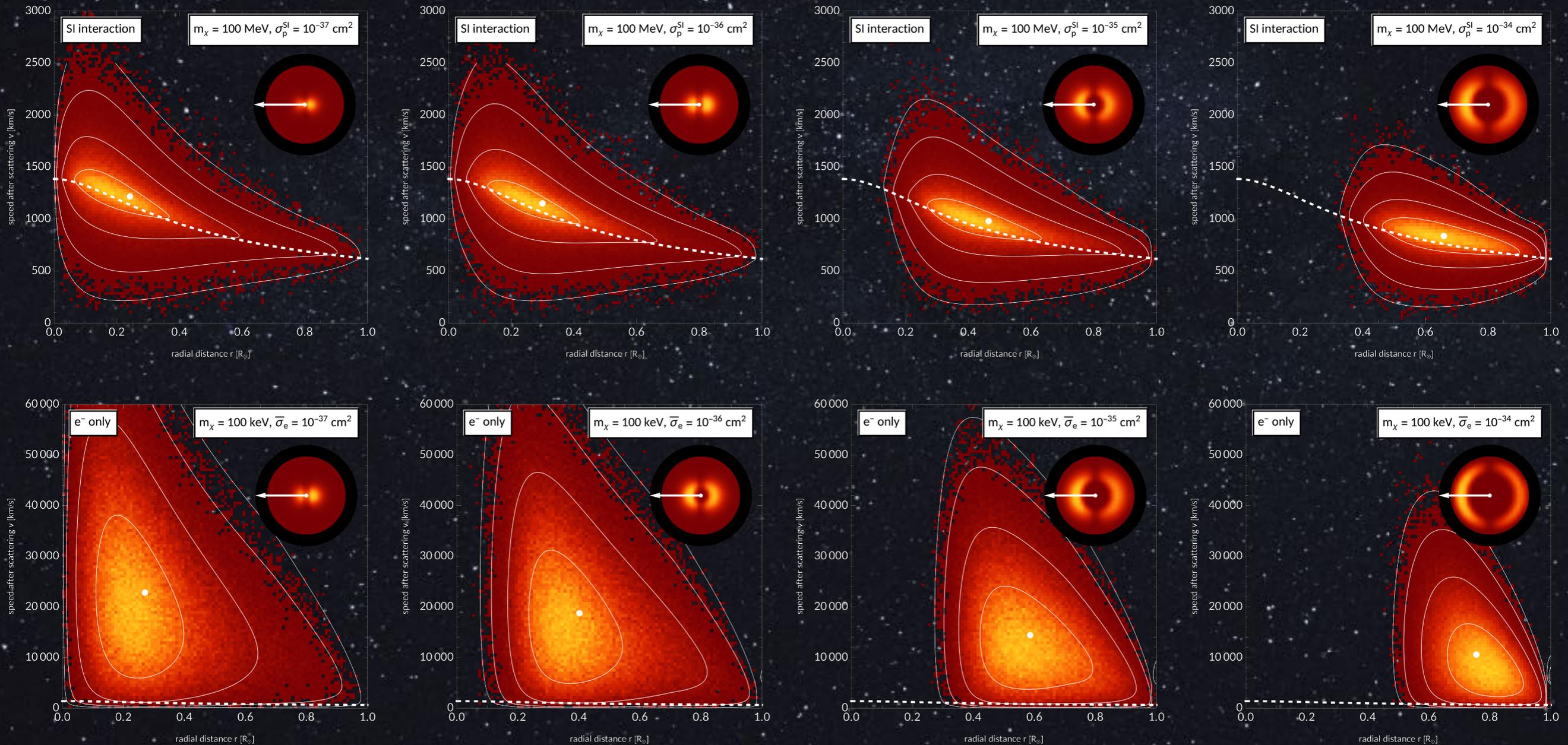




# The first scattering inside the Sun

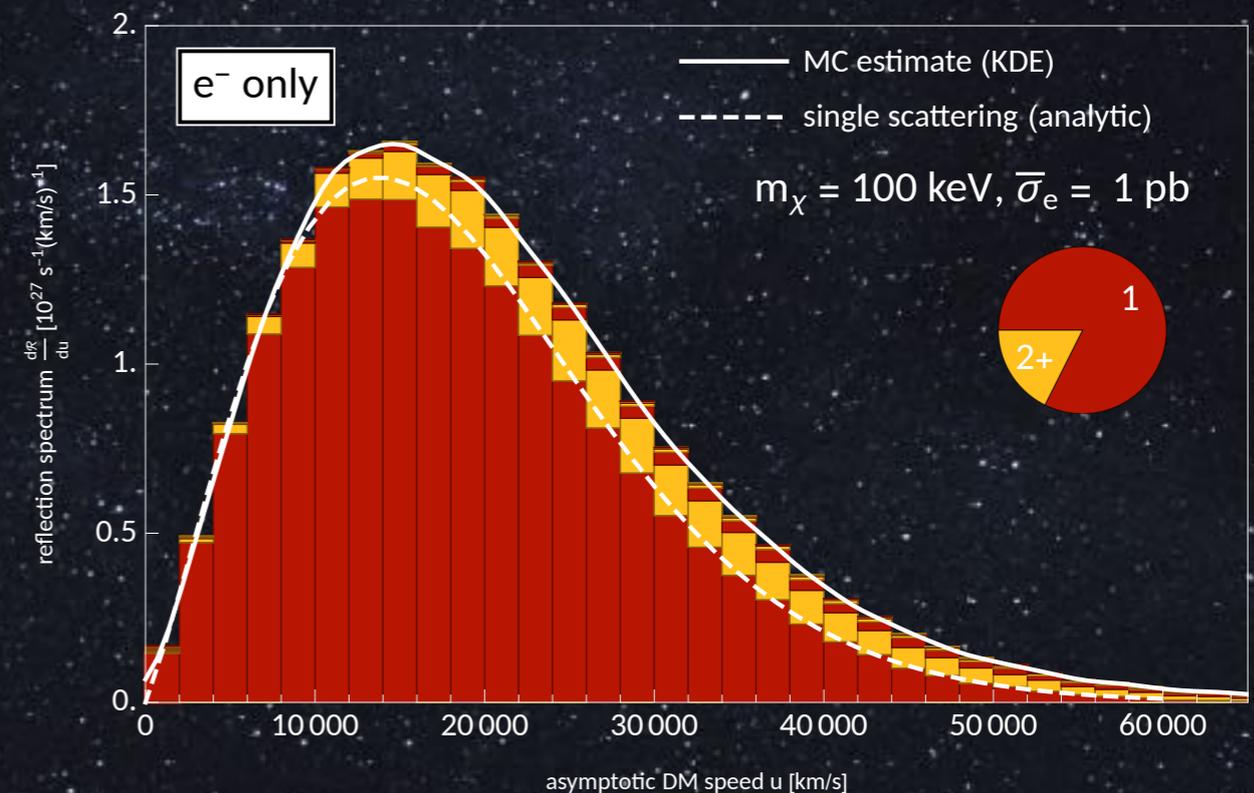
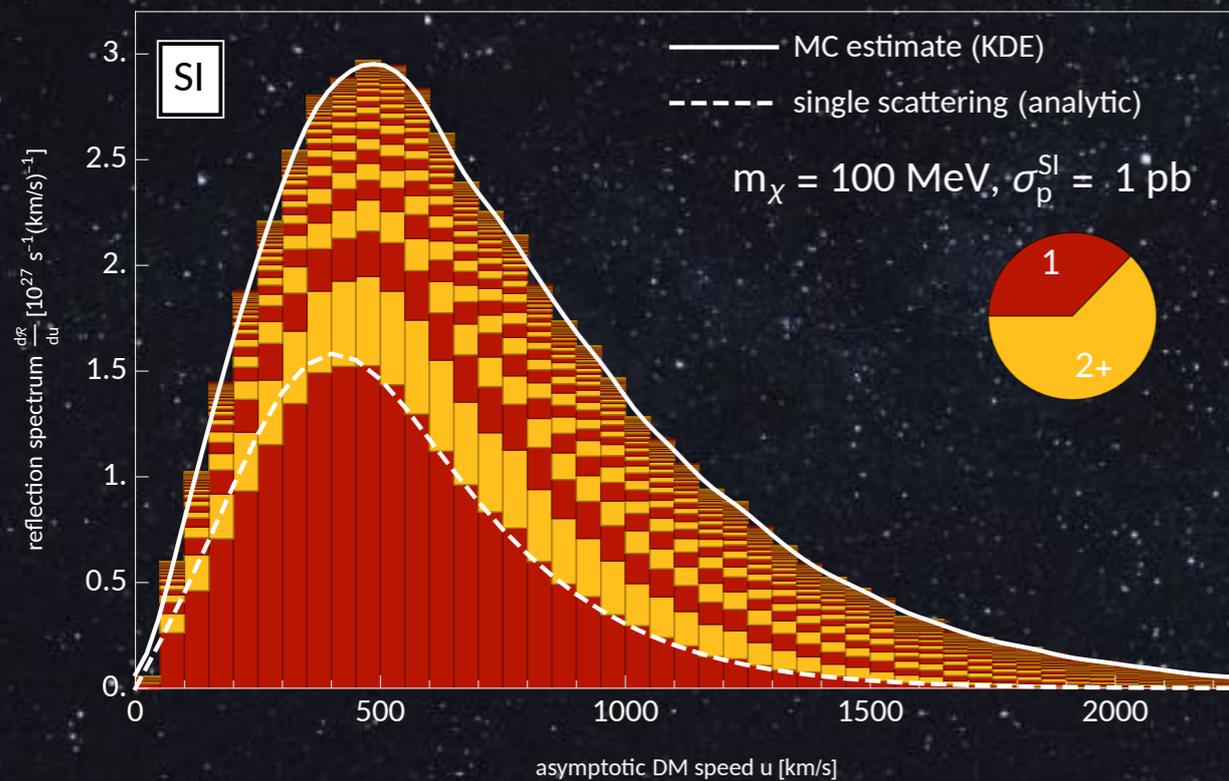
- The first scattering allows comparison with our analytic description of DM scatterings inside the Sun.

TE, C. Kouvaris, N. G. Nielsen, PRD, 97 (2018), 063007



# The impact of multiple scatterings

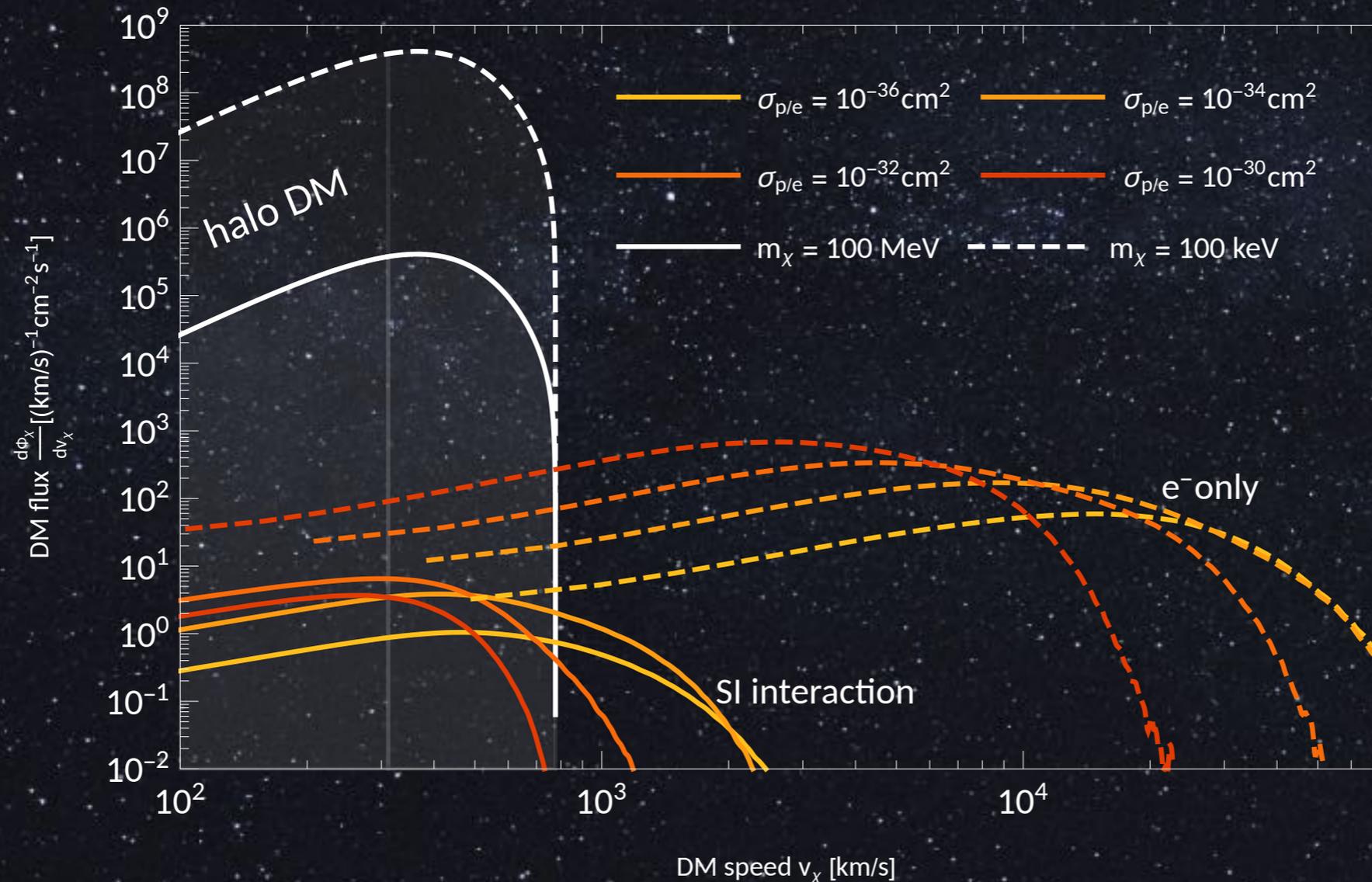
- Using MC simulations allows to quantize the contribution of multiple scatterings to the SRDM flux.



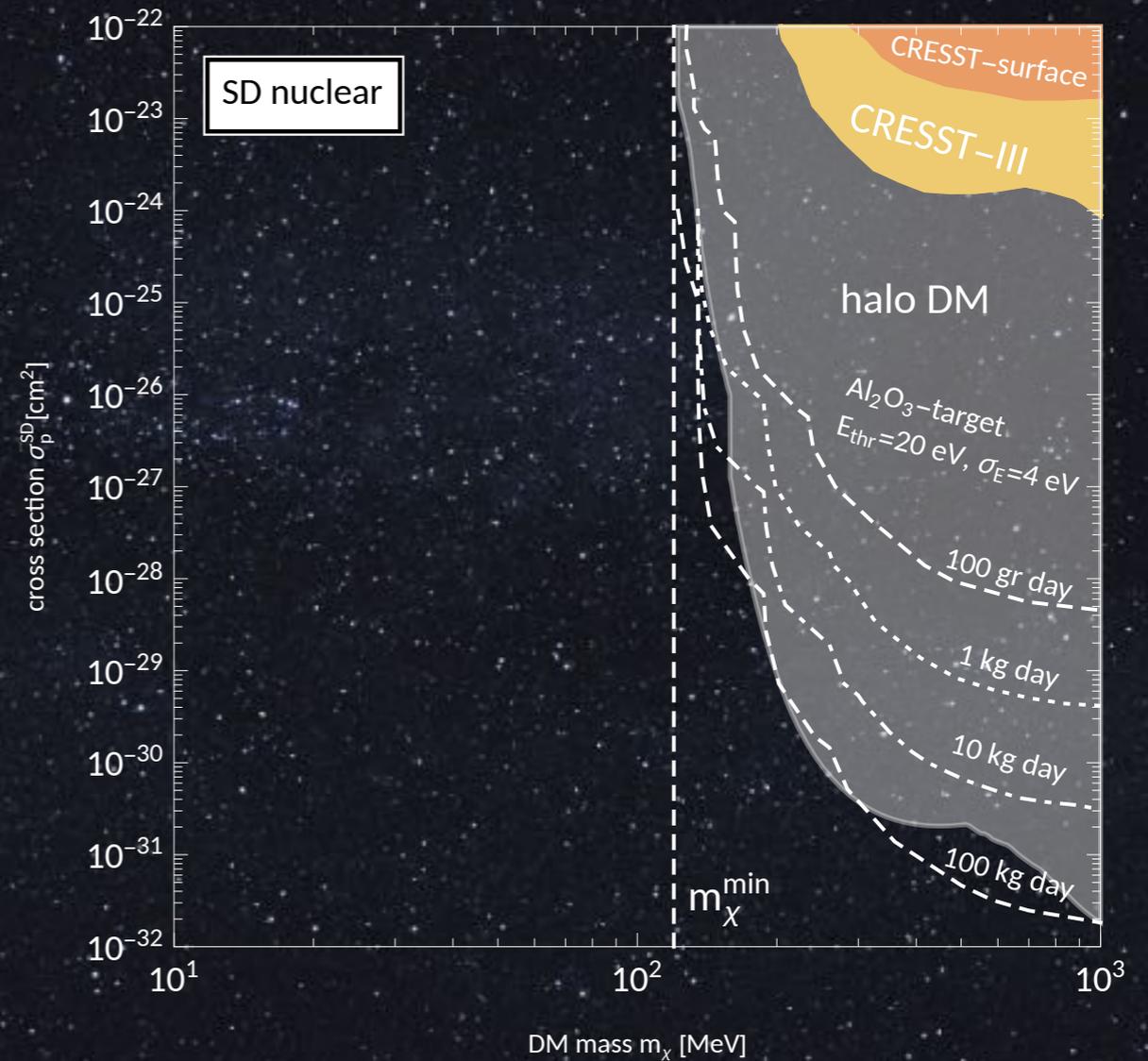
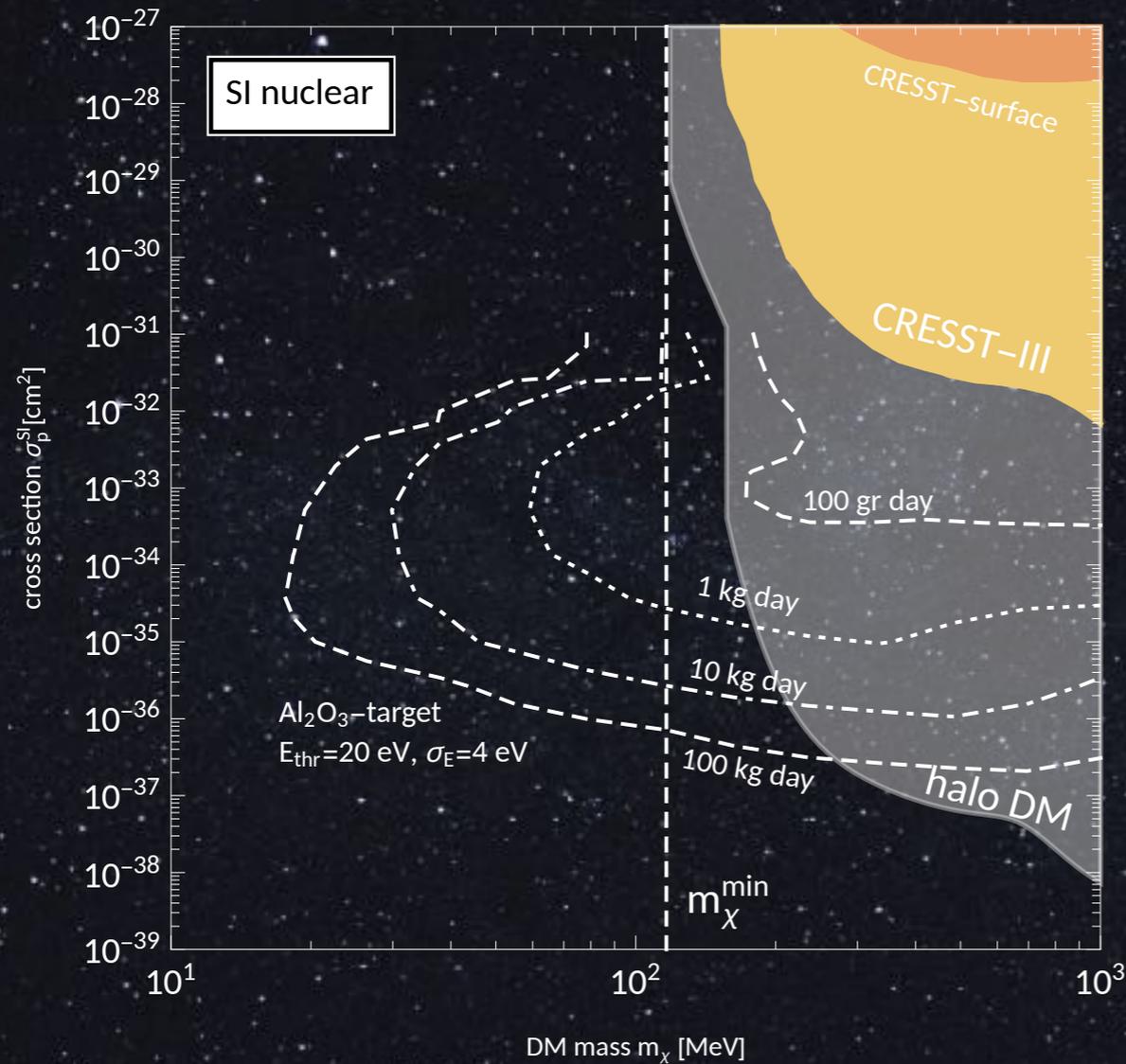
➔ Great consistency between analytic formalism and MC description!

# The solar reflection DM flux

- Compare the SRDM flux to the halo DM flux.



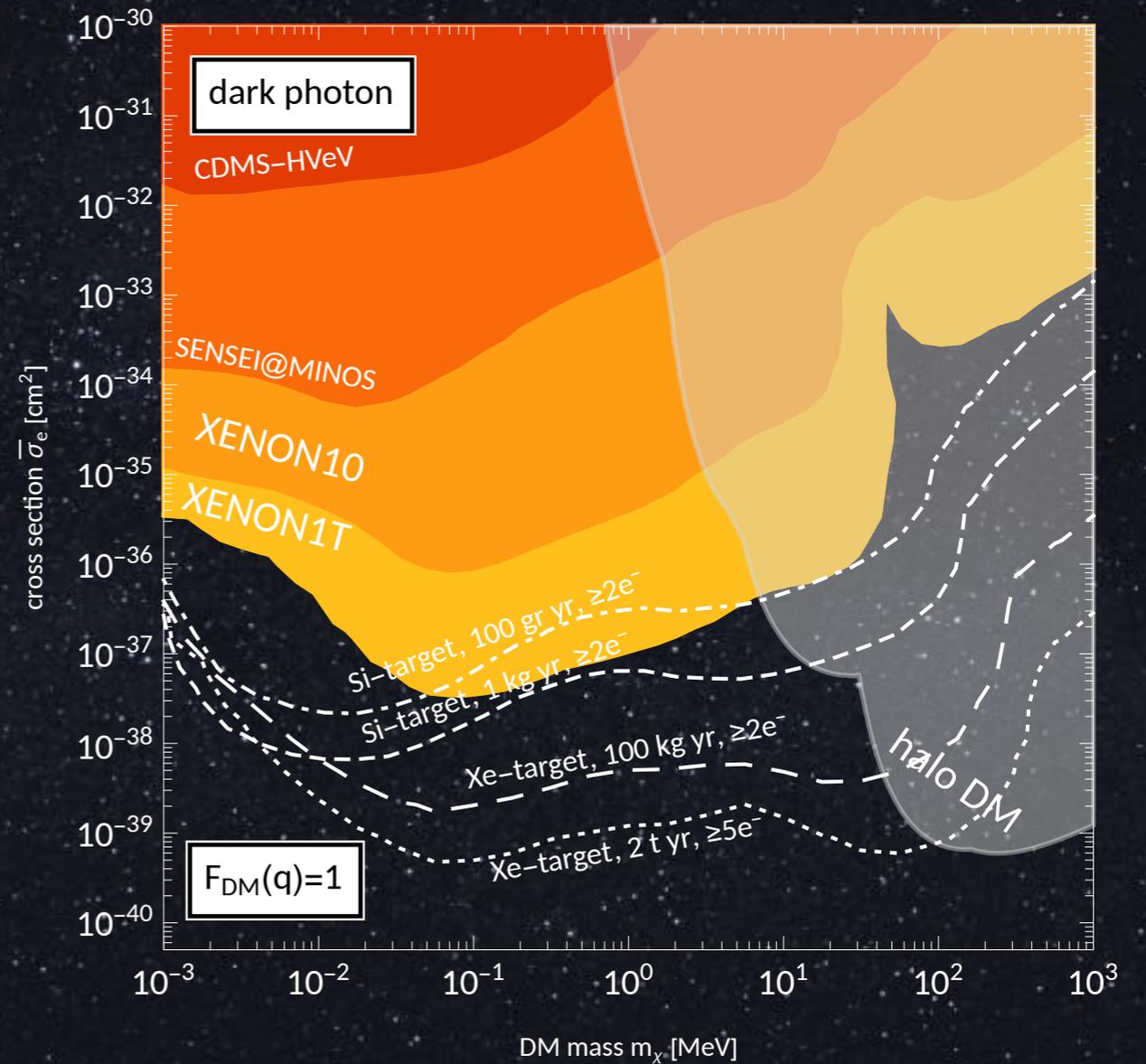
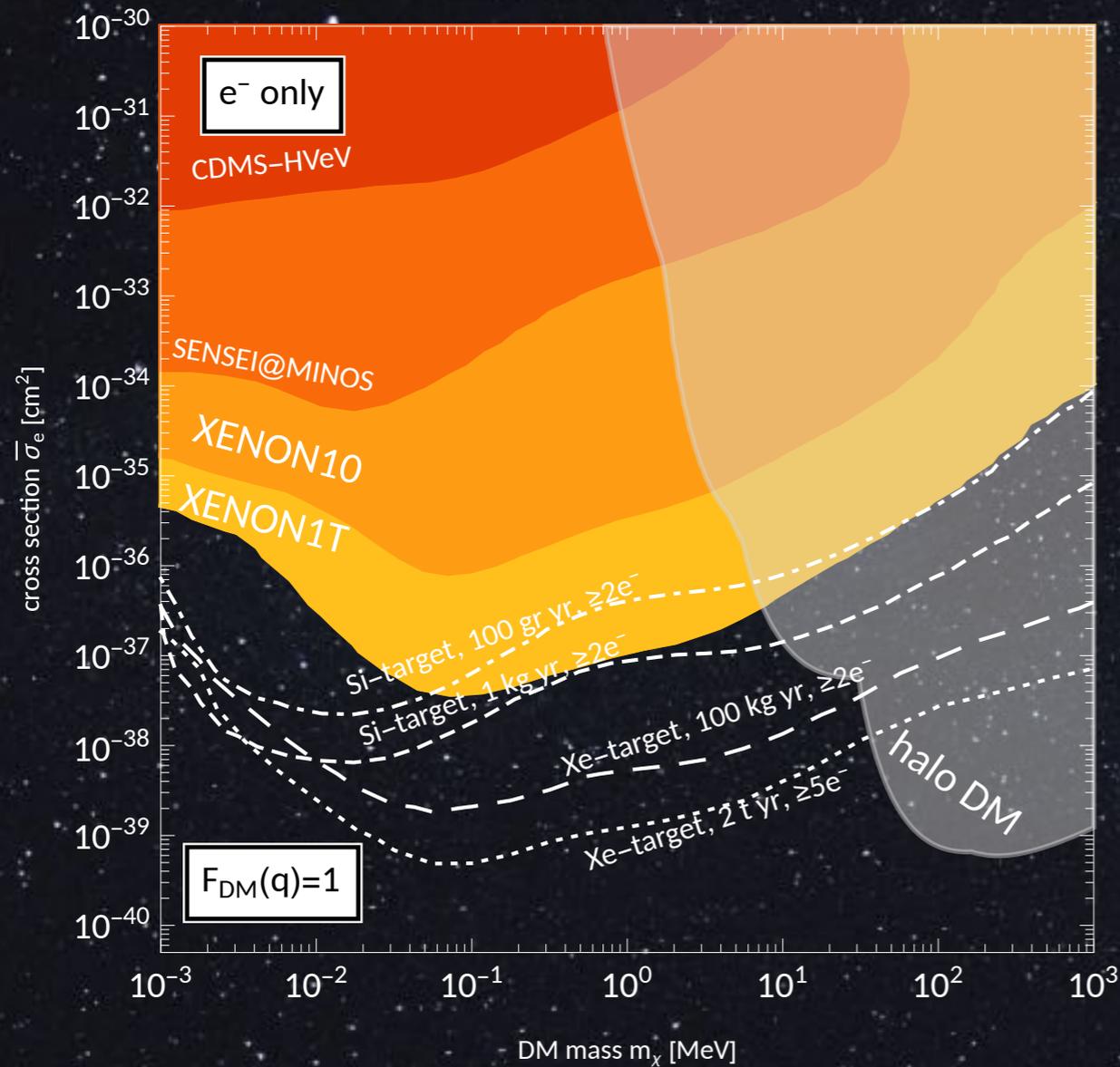
# Exclusion limits for DM-nucleus interactions



➔ Higher exposures probe lower masses!

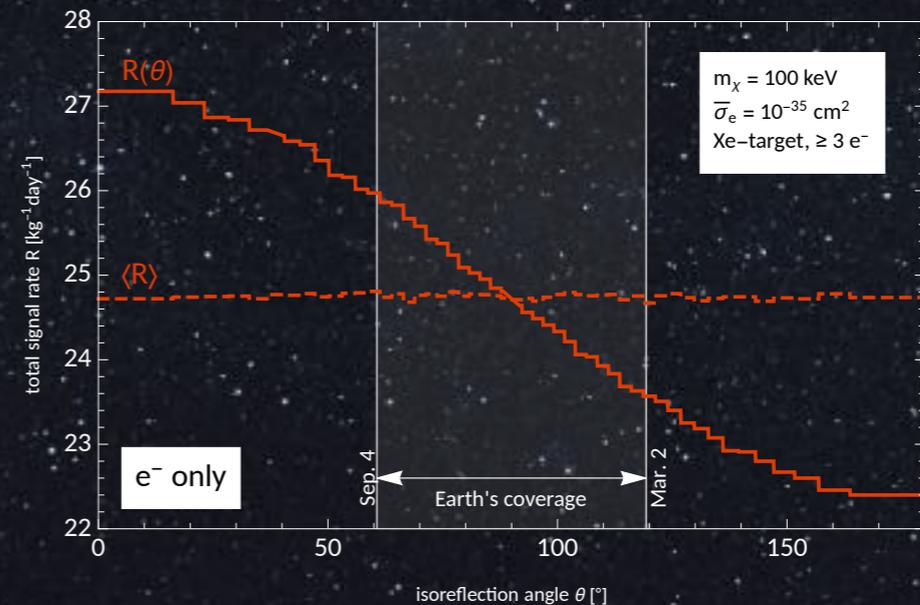
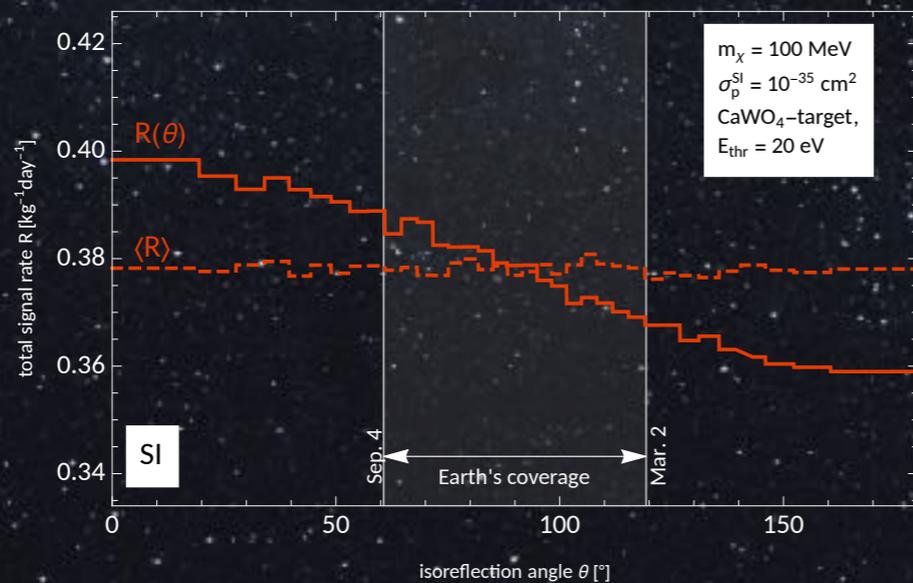
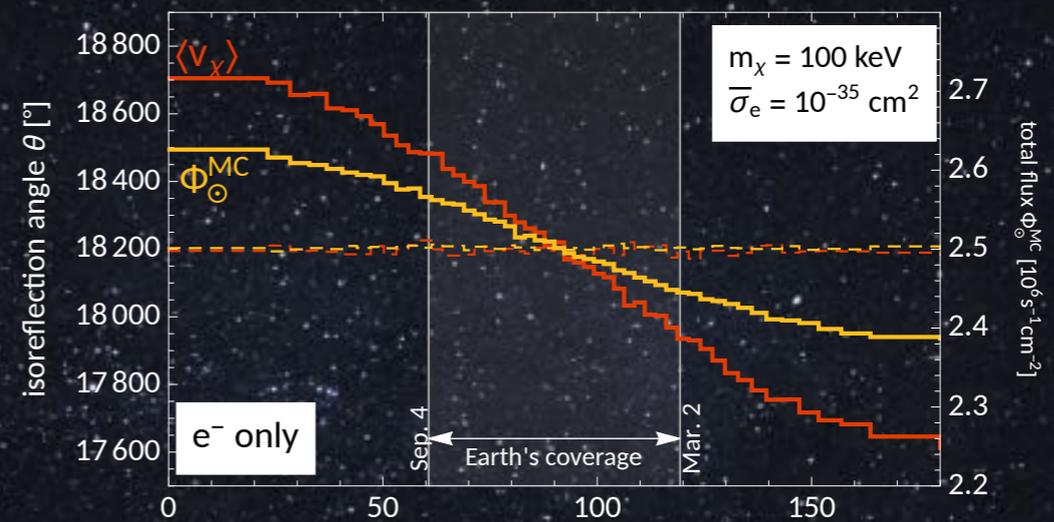
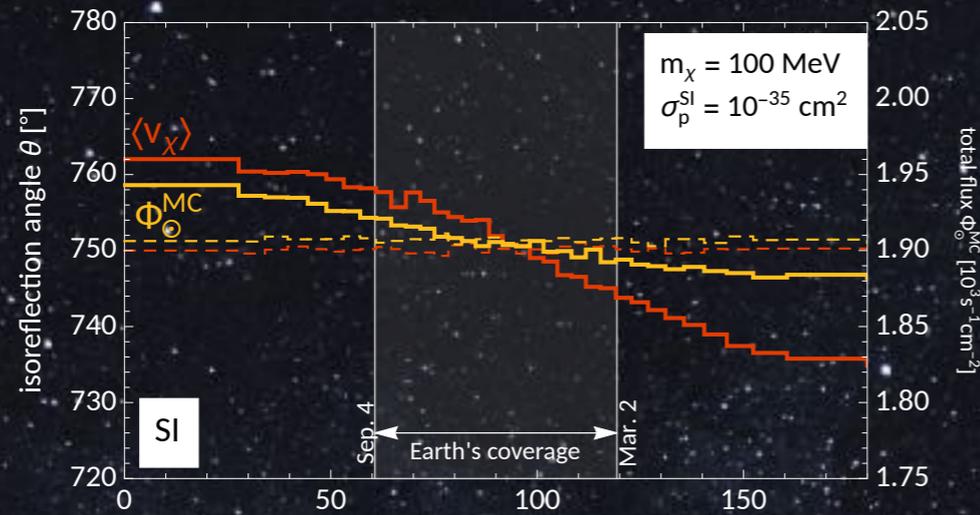
- For SI interactions, next generation experiments can extend their sensitivity to lower masses through SRDM.

# Exclusion limits for DM-electron interactions



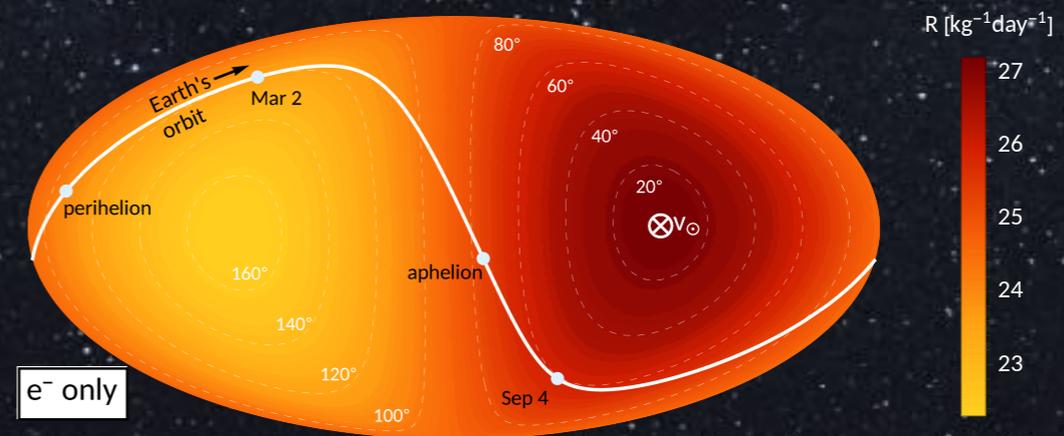
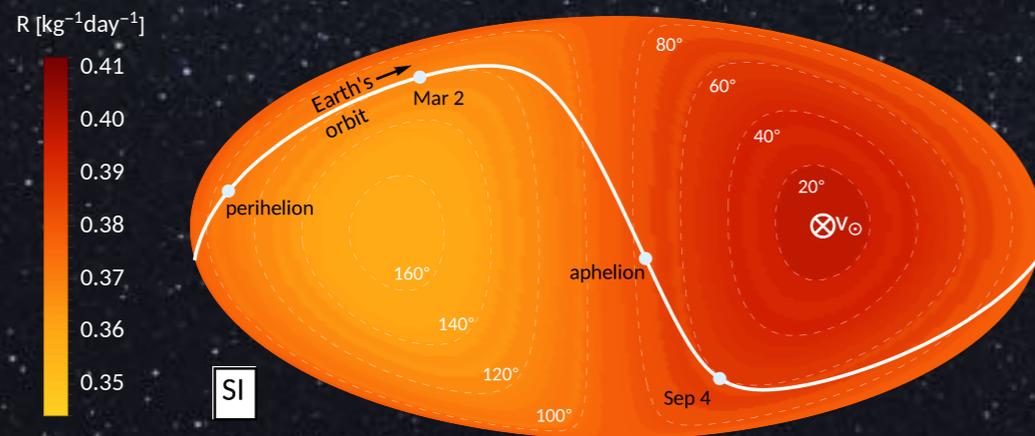
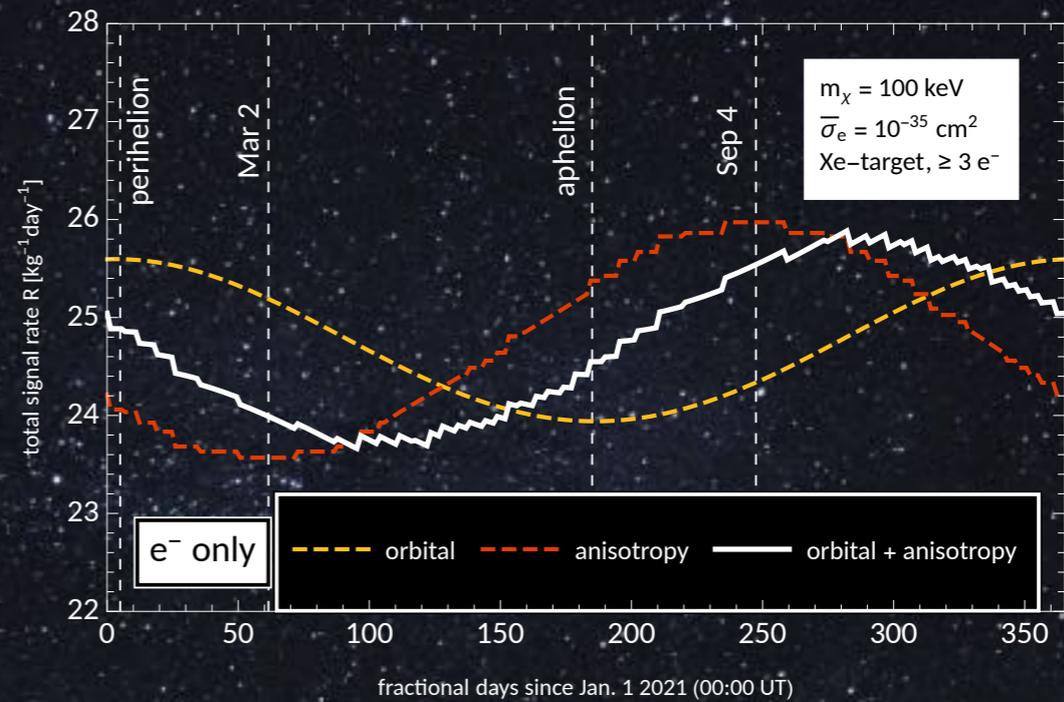
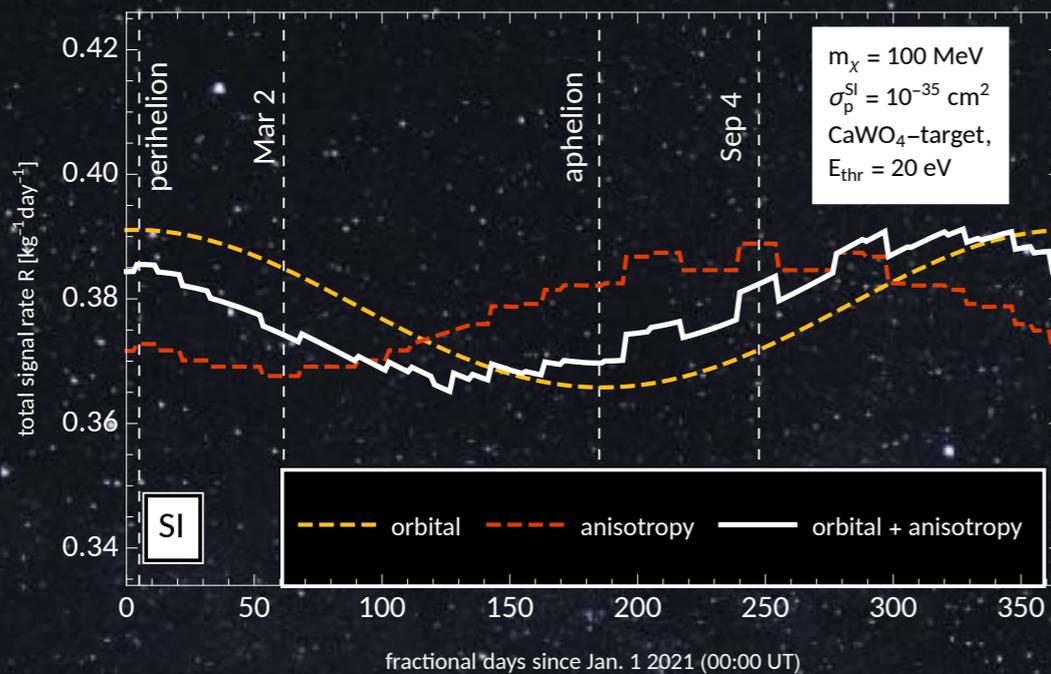
→ Solar reflection extends the experiments' mass sensitivity by multiple orders of magnitude.

# Anisotropies of the SRDM flux



➔ The Sun does **not** eject SRDM particles isotropically into the solar system.

# Annual signal modulations



➔ The orbital and anisotropy modulations feature comparable amplitudes.

# V. Summary & Outlook

- Scatterings on hot solar targets can accelerate light DM particles.
- The energetic flux of reflected DM particles can extend the sensitivity of direct detection experiments to lower masses especially for DM-electron interactions.
- We performed MC simulations to describe the SRDM flux for 4 different DM scenarios and derived exclusion limits based on existing and planned experiments.
- An SRDM signal would feature a rich modulation signature. We investigated the annual modulation.



- So far we only considered heavy mediators. What about light mediators?
- Migdal scatterings + SRDM could potentially extend sensitivity to low mass DM searches via nuclear recoils.

```
double v2 = sqrt(GNewton*msun/p)*(e*sin(theta2)*x2.normal2d,
// double F2 = ncosh((e+cos(theta2))/(1.0+e*cos(theta2)))) // double M2 = asinh(F2)-F2; // double t2 = sq
stance at which we sample from the halo distribution double R=100.0*AU; if(R<100.0*rSun) { cerr <<"Warning in
holders[1],vEarth); double u = Rejection_Sampling(pdf,0.0,(vesc+vEarth),ymax,PRNG); //Velocity Direction
om Point on a flat disk at distance R Eigen::Vector3d az=vini.normalized(); Eigen::Vector3d ox(0,az/2
cos(phi)+e*cos(phi)+ey); // cout <<"Norm = "<<xini.norm()/AU<<endl; // cout <<"Norm = "<<(R+ex, norm)/A
[0]/rSun<<"\t"<<IC.Position()[1]/rSun<<"\t"<<IC.Position()[2]/rSun<<"\t"<<IC.Radius()/rSun<<"\t"<<IC.Velocity(
r_Shift(IC,rMax,model); } //2. Orbit simulation //right hand sides of the 1st order equations of motion, double
Euler CRUR // void EC_Step(double dt,double &r,double &v,double &phi,double &J,double dt,Sun::Model &model) // {
// //RK coefficients: // double k_r[4]; // double k_v[4]; // double k_p[4]; // k_r[0]= dt*drdt(v); // k_v[0]
= dt*dvdt(r+k_r[1]/2.0,J,model); // k_p[2]= dt*dphidt(r+k_r[1]/2.0,J); // k_r[3]= dt*drdt(v+k_v[2]
2)+k_v[3]); // phi+= 1.0/6.0*(k_p[0]+2.0*k_p[1]+2.0*k_p[2]+k_p[3]); // //ODE integration with Runge Kutta Fehl
k_v[0]= dt*dvdt(r,J,model); k_p[0]= dt*dphidt(r,J); k_r[1]= dt*drdt(v+k_v[0]/4.0); k_v[1]= dt*dvdt(r+k_r[0]/
.0+k_r[1],J); k_r[3]= dt*drdt(v+1932.0/2197.0*k_v[0]-7200.0/2197.0*k_v[1]+7296.0/2197.0*k_v[2]); k_v[3]= dt*
.0*k_v[1]+3680.0/513.0*k_v[2]-845.0/4104.0*k_v[3]); k_v[4]= dt*dvdt(r+439.0/216.0*k_r[0]-8.0*k_r[1]
-3594.0/280.0*k_v[2]+1859.0/4104.0*k_v[3]-11.0/40.0*k_v[4]); k_v[5]= dt*dvdt(r-8.0/27.0*k_r[0]-2.0*k_r[1]-3M
ble r4= r+28.0/216.0*k_r[0]+100.0/288.0*k_r[2]+2107.0/4104.0*k_r[3]-1.0/8.0*k_r[4]; double v4= v+28.0/216.
k_r[2]+28561.0/56430.0*k_r[3]-9.0/50.0*k_r[4]+2.0/55.0*k_r[5]; double v5= v+16.0/135.0*k_v[0]+6656.0/12025.0*k
r4),abs(v5-v4),abs(phi5-phi4)); std::vector<double> tolerance = {1.0*km,1e-3*km/sec,1e-6}; //New steps:ie
s),std::end(deltas)); //Next steps if(err[0]<tolerance[0]&&err[1]<tolerance[1]&&err[2]<tolerance[2]) { if(r<
xi -= speed*dt/vcr; } t+= dt; r = r4; v = v4; phi = phi4; dt = std::min(dtmin, std::min(delta*dt,dtMax)); } else { dt
vector<Eigen::Vector3d>& axes) { double v_phi = J/pow(r,2); Eigen::Vector3d xNew = r*(cos(phi)*axes[0]+sin(phi)*ax
scattering or (b) leaving the sun. bool Propagate(Event& x0,DM_Particle& DM,Sun::Model& model,std::m
)).normalized(); Eigen::Vector3d ax_y = ax_z.cross(ax_x); std::vector<Eigen::Vector3d> axes = {ax_x,ax_y
f)).dot(ax_z); //Sample -log(1-xi) double logXi = -1.0*log(1.0-ProbabilitySample(PRNG)); //Start integ
_step(t,r,v_r,phi,J,dt,logXi,DM,model); if(Mxi0==0) { Event xNew =PlaneTo3D(t,r,phi,v_r,J,axes); t <<xNew.Posit
eed(model.vEsc2(xNew.Radius())) <<"\t" <<InUnits(xNew.Speed(),km/sec) <<endl; } // double dd = (xOld.Position
num // t <<xNew.Position()[0]/rSun<<"\t"<<xNew.Position()[1]/rSun<<"\t"<<xNew.Position()[2]/rSun<<"\t" <<t/sec
Old<rMax&&rMax&&(v_r+v_r+J)/r/r>model.vEsc2(r)) { propagate=false; x0=PlaneTo3D(t,r,phi,v_r,J,axes); } else
(Event& x0,DM_Particle& DM,Sun::Model& model,std::mt19937& PRNG) { // double speed0=x0.Speed(); // cout <<"Scatt
dinate{1.0,ThetaSample(PRNG),PhiSample(PRNG)}; double vRel = (vTarget-x0.Velocity()).norm(); Eigt
eed0<<endl; // if(speed0>sqrt(model.vEsc2(r))&&vnew.norm()<sqrt(model.vEsc2(r))) cout <<"\tcapture"<<endl; //
ol& model, unsigned int &nScattering, std::mt19937& PRNG) { //Save trajectory of stream
x.Speed()*x.Speed()-model.vEsc2(x.Radius())/E0<<endl; //Output // Event x = IC; bool success; //Counters nScat
if(r<rSun) { if(nScattering>nScattering_max) { success = false; break; } // double E1=DM.mass/2.0*(x
*(x.Speed()*x.Speed()-model.vEsc2(x.Radius())/E0<<endl; // if(E1>0 && E2<0) { // cout <<"capture"<<endl;
<"\t" <<r/rSun << endl; // f. close(); // // } // if(x.Speed() < sqrt(model.v
bolution()[0]/rSun<<"\t"<<xFinal.Position()[1]/rSun<<"\t"<<xFinal.Position()[2]/rSun<<"\t"<<xFinal.Radius()
ces=false; if(x.time()/sec<9999) return false; return success; } Result Generate_Data(Configuration& config
r,numprocs-1)? 0 : config.rank-1; int tag = 0; MPI_Status status; MPI_Request send_request; // Datapoin
data; data.resize(config.nRings); //Counters //Total number of simulated particles unsigned long int G_Coun
vector<unsigned long int> L_Counter_Data_new(config.nRings,0); //Number of passes // std::vector<unsigned
ber of scatterings; std::vector<double> G_nScattering_Av(config.nRings,0.0); std::vector<double> L_nSca
Isend(&G_Counter_Data.front(),config.nRings,MPI_UNSIGNED_LONG,destination,tag,MPI_COMM_WORLD
G)) unsigned int nScattering=0; bool success = Simulate_Trajectory(x,DM,model,nScattering,PRNG);
_nScattering_Av[Isorting]+nScattering)/L_Counter_Data[Isorting]; double speed=x.Speed(); t[nS
_DHM_WORLD,&flag,&status); if(flag) { //Receive the tokens MPI_Recv(&G_Counter_Data.front(),config.nRi
r_Data[1]=L_Counter_Data_new[1]; L_Counter_Data_new[1]=0; } // cout <<config.rank <<"\t" <<G_Coun
else if (nRings>config.nData) tag = status.MPI_TAG; //Pass on the tokens, unless you are the very la
gth cout <<"\r"; for(int i=0;i<2.0*BarLength;i++)cout <<" "; cout <<"\r"; for(int i=0;i<BarLength
) cout <<"\r"; for(int j=0;j<10*BarLength;j++) cout <<" "; cout <<"\r"; } // for(int i=0
_Counter_Free,1,MPI_UNSIGNED_LONG_LONG,MPI_SUM,MPI_COMM_WORLD); MPI_Allreduce(&L_nSca
> Global_Data; for(int i = 0;i<config.nRings;i++) { unsigned long int Global_SampleSize; MPI
cy_displs[config.numprocs]; MPI_Allgather(&L_Counter_Data[i],1,MPI_UNSIGNED_LONG,&nData_Loc
SampleSize); // MPI_Gatherv(&nData[i].front(),data.size(),mpi_datapoint,&ring_Data.front(
) MPI_Barrier(MPI_COMM_WORLD); // for(int i=0;i<config.nRings;i++) // { // if(config.nR
h,1,MPI_DOUBLE,MPI_MAX,MPI_COMM_WORLD); return Result(Global_Data,DM,G_Counter_Simulat
std::vector<double>& nscav,double dt) { data=dat; particle = p; // sample_size
} nscattering_mean=nscav; nscattering_mean_tot=0.0; for(unsigned int i=0;i
n &config) { if(config.rank==0) { std::cout <<"Simulation summa
<Round(1.0*nSimulations/duration)<<"(sec)"<<std::endl; if(config.nRings==1) st
<" " <<Round(100.0*nFails/nSimulations)<<"%"<<std::endl <<"\tDuration[s]:"
ize[i] <<"\t\t\t" <<Round(nScatterings_mean[i])<<std::endl; } } } #inc
ructions.cpp using namespace libconfig; //1. Struct with input param
nData=nD; //Parallelisation rank = myrank; numprocs= np; } Configur
const fileIOException &fileex) { std::cerr <<"I/O error while re
try { size_t = cfg.lookup("simID").c_str(); } catch(const
file." << endl; exit(EXIT_FAILURE); } // Isort
" << endl; } //2. Event
```

Thank you!

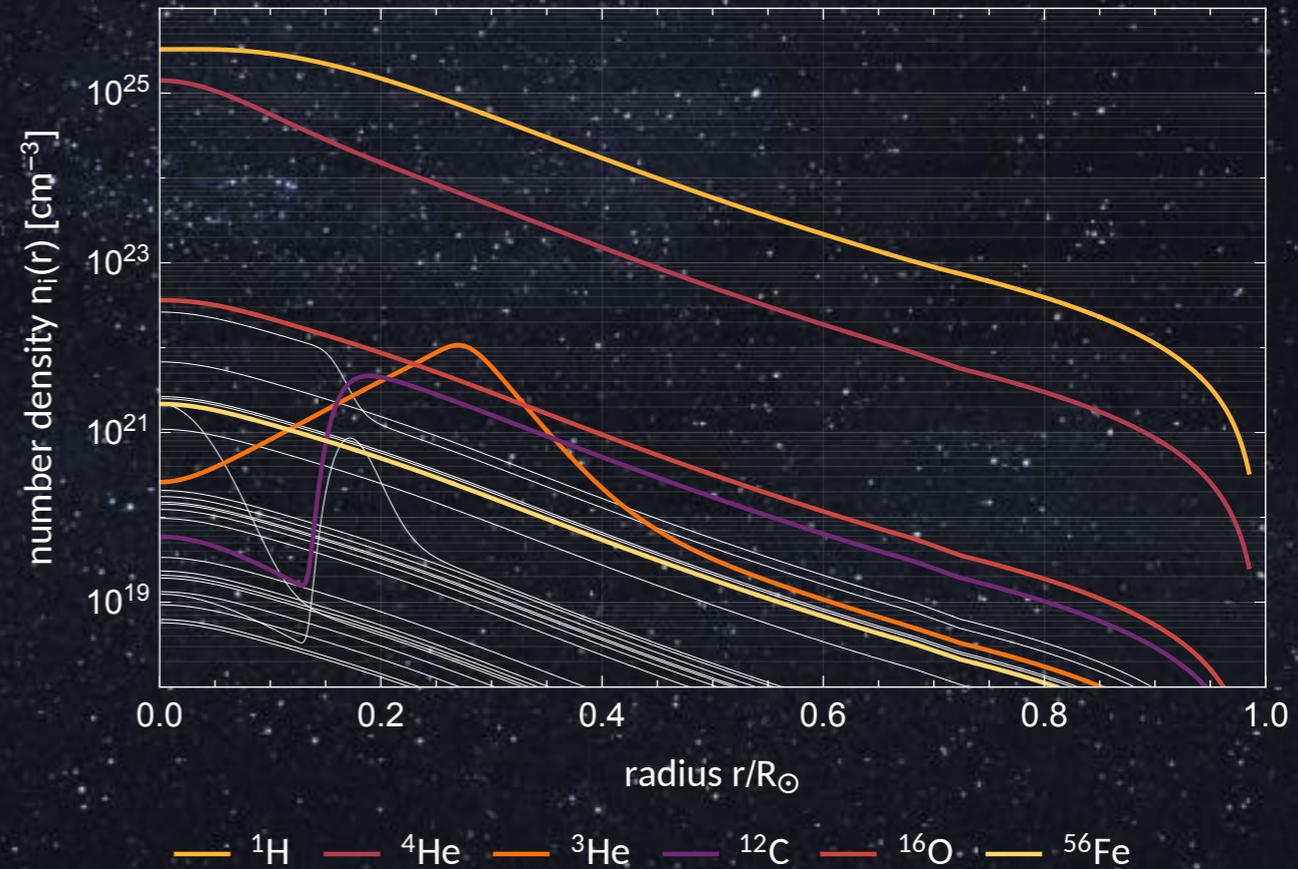
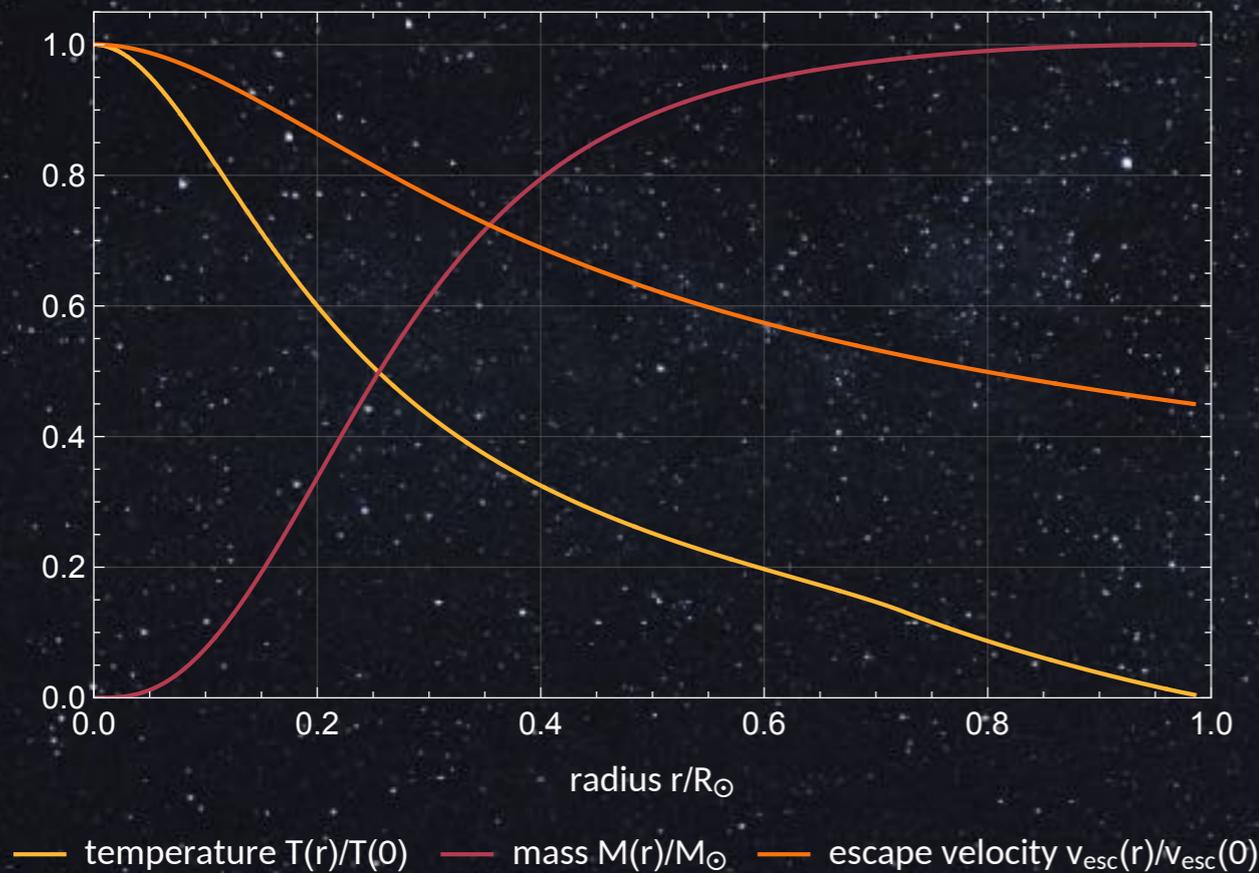
# VI.

## Backup Slides

# The solar model

Standard Solar Model AGSS09

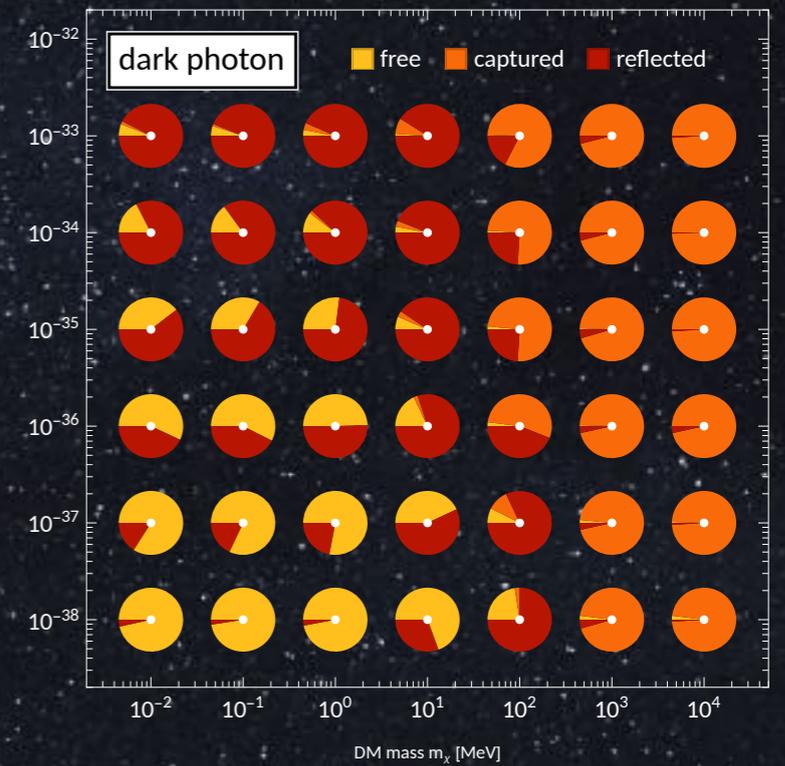
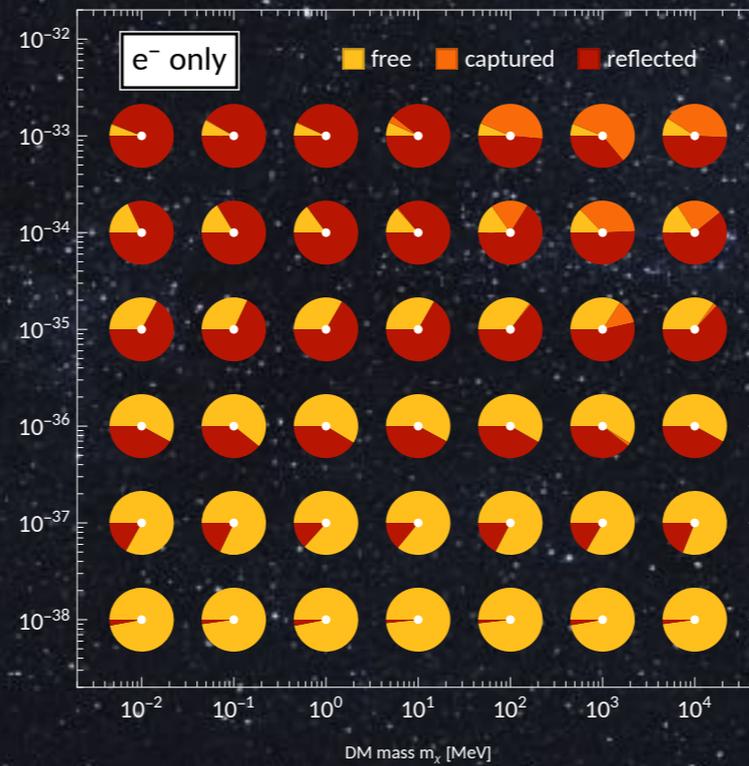
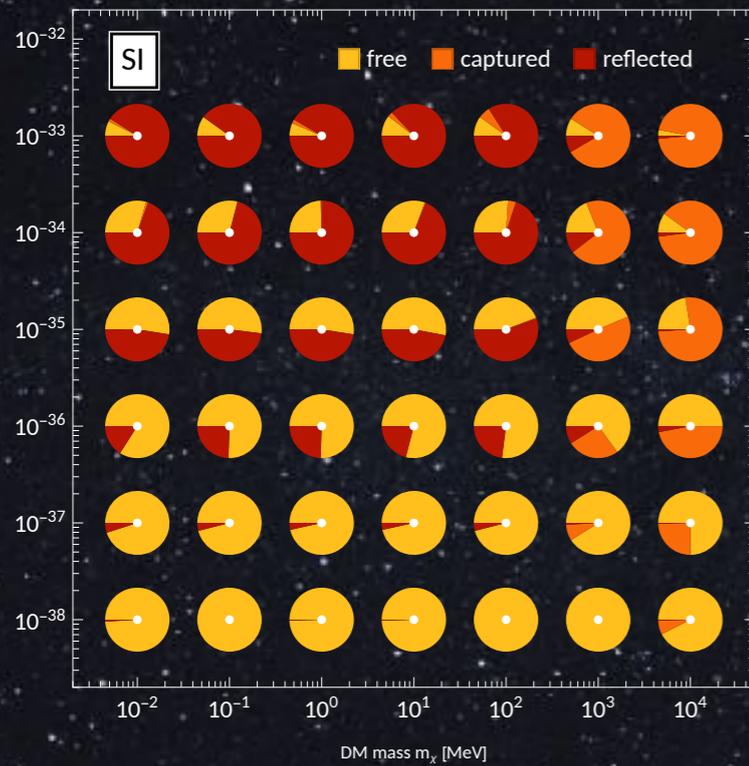
A. Serenelli et al., *Astrophys. J.* 705 (2009), L123-L127



- Mass Function  $M(r)$
- Temperature  $T(r)$

- Nuclear composition
- Number densities

# DM Gravitational capture rate



# Runge-Kutta-Fehlberg (RK45)

E. Fehlberg, NASA technical report, NASA, 1969

Adaptive method for the numerical solution of 1st order ODEs.

$$\dot{y} = f(t, y), \quad y(t_0) = y_0,$$

Requires the same number of function evaluation of RK6, but is rather a combination of RK4,

$$y_{k+1} = y_k + \frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4101}k_4 - \frac{1}{5}k_5,$$

and RK5,

$$\tilde{y}_{k+1} = y_k + \frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6.$$

The comparison of both yields an estimate of the error, and allows to choose the step size adaptively.

$$\Delta t_{k+1} = 0.84 \left( \frac{\epsilon}{|y_{k+1} - \tilde{y}_{k+1}|} \right)^{1/4} \Delta t_k$$