Photovoltaic Dynamic Material Flow Assessment Model

PV DMFA

Release 04.04.2022

User Manual Documentation

Authors:

Sherif A. Khalifa, Benjamin Mastrorocco, Dylan Au

Project Leaders:

Alberta P. Carpenter, Teresa M. Barnes, Jason B. Baxter

Produced: March 2022

Acknowledgements

This work draws from many areas of expertise particularly the PV semiconductors, sustainability, and Industrial ecology fields. Authors thank Margret Mann for her support in initiating the project and thank Scott Nicholson for his insightful feedback in the initial stages of building the spreadsheet model that later migrated into open-source Python language. Special thanks to Garvin Heath at NREL for providing critical feedback on the PV recycling markets. Authors also thank Silvana Ovaitt and Heather Mirletz for useful discussions; PV DMFA and PV ICE models were developed in close collaboration between authors. Authors also thank all contributing personnel in the Advanced Manufacturing Office (AMO) at NREL for their feedback on improving their work, particularly Samantha Reese.

License:

Software Citation:

Khalifa SA, Mastrorocco BV, Au DD,et al. Dynamic material flow analysis of silicon photovoltaicmodules to support a circular economy transition.ProgPhotovolt Res Appl. 2022;1-22. doi:10.1002/pip.3554

Table of Contents

1. Introduction

The Photovoltaic Dynamic Material Flow Assessment Model (PV DMFA) is a materials systems framework designed to track/trace the material use in solar PV utility sector from resource extraction (i.e. cradle) to waste management and material circularity practices (i.e. grave) in the 21st century. The model is based upon end use electricity generation. The model can be leveraged to estimate the magnitude of a given PV material flow in any stage in its life cycle and provide a feedback on material flows in PV economy in response to changes in a variety of input parameters that influence primary material acquisition, material exchange with non-utility economy sectors, material production and assembly, material use in PV panels and subsequent impacts of various circularity strategies on the quality and quantity of materials. The model is designed to provide sustainability feedback for people involved in the PV value chain:

- PV researchers
- Manufacturers and supply chain vendors
- Electronics waste management industry
- PV technology developers
- Policy analysts, project managers and engineers

1.1 What this tool can answer?

PV DMFA is a comprehensive material sustainability tool that accurately applies life cycle assessment (LCA) method as detailed in ISO codes 14040/14044. Although carrying out an extensive environmental impact assessment results is outside the scope of this model, the results from this model can be a valuable resource for a wide variety of stakeholders particularly LCA practitioners and technoeconomic researchers. Particularly, the quantity and quality of material flows can be further analyzed to obtain the associated energy and emissions bills to perform a complete LCA study. This tool can provide estimations for waste generation, raw material extraction, production losses to landfill/recycled/remanufactured, material savings from post use waste management and material compositional changes during EOL handling and metallurgy for any modeled changes in PV module design, material value chain or waste management.

Some of the most important questions PV DMFA can answer are:

- Identification of sensitive PV-specific and material-specific parameters that drive waste generation and primary material extraction.
- Comparison between various material circularity strategies on waste generation and material savings in the PV value chain.

- Analysis of the contribution of the life cycle stages to the overall waste and associated environmental loads.
- Quantifying the impacts of different PV module design decisions on the PV material value chain.
- Quantifying material demand in response to changes in electricity demand.
- A basis for conducting formal LCA and techno-economic assessment studies to evaluate the environmental footprint of PV deployment.

In this document, the model framework and data structure are presented to guide users to understand the underlying calculation sequence and allow them to run the model to obtain results.

1.2 Installation

In order to run the model, the user must download Anaconda Navigator that includes **Juypter Notebook**. Anaconda Navigator is a program that will allow the user to download additional packages that are required for the model to run.

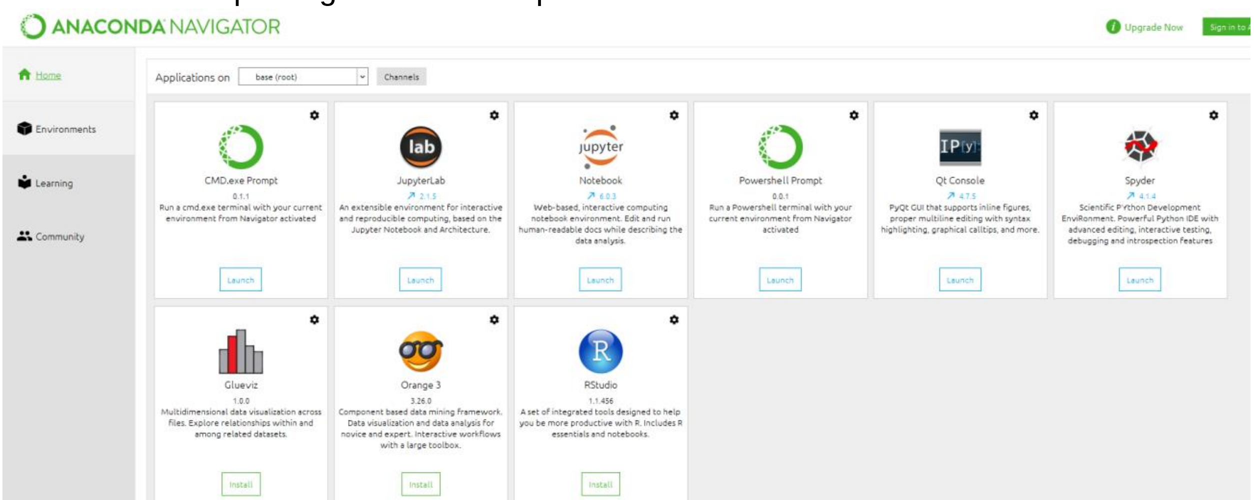


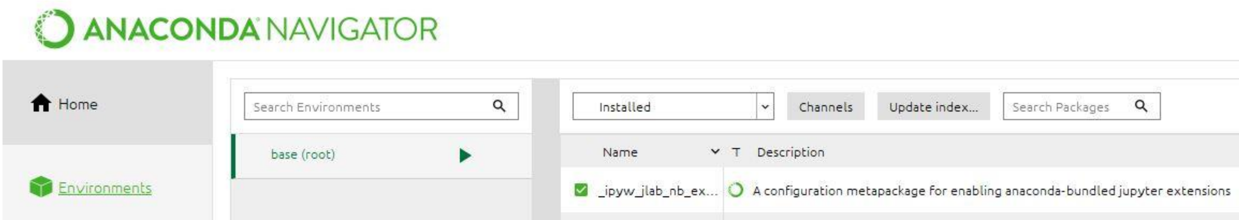
Figure 1: Anaconda Navigator user interface

Once Anaconda Navigator is downloaded, launch the program, which will look the image in Figure 1. In addition to Juypter Notebook, the required packages need to be installed:

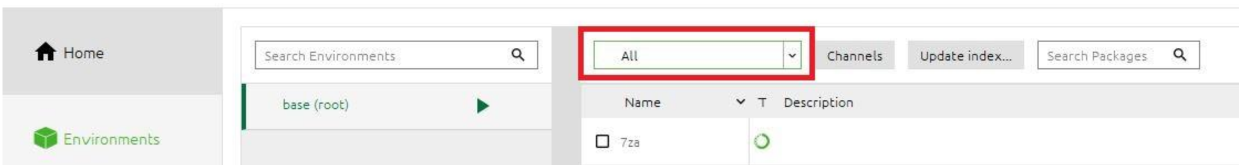
- numpy
- pandas
- matplotlib
- openpyxl
- plotly

In order to install these:

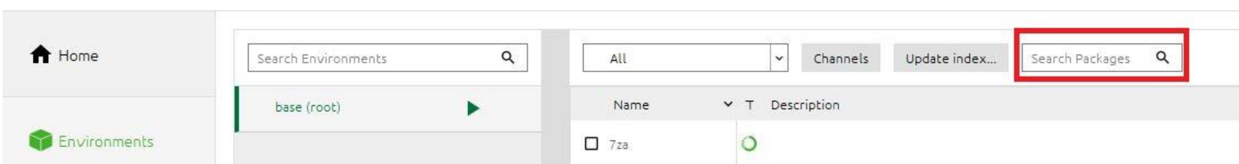
1. Move to the environment tab where all packages can be found and installed.



2. Change the drop down to All as shown below:



3. Search the packages in the search bar as shown below:



4. Install the packages if they are not already installed by default.

2. Getting Started

2.1 An Electricity Generation Model

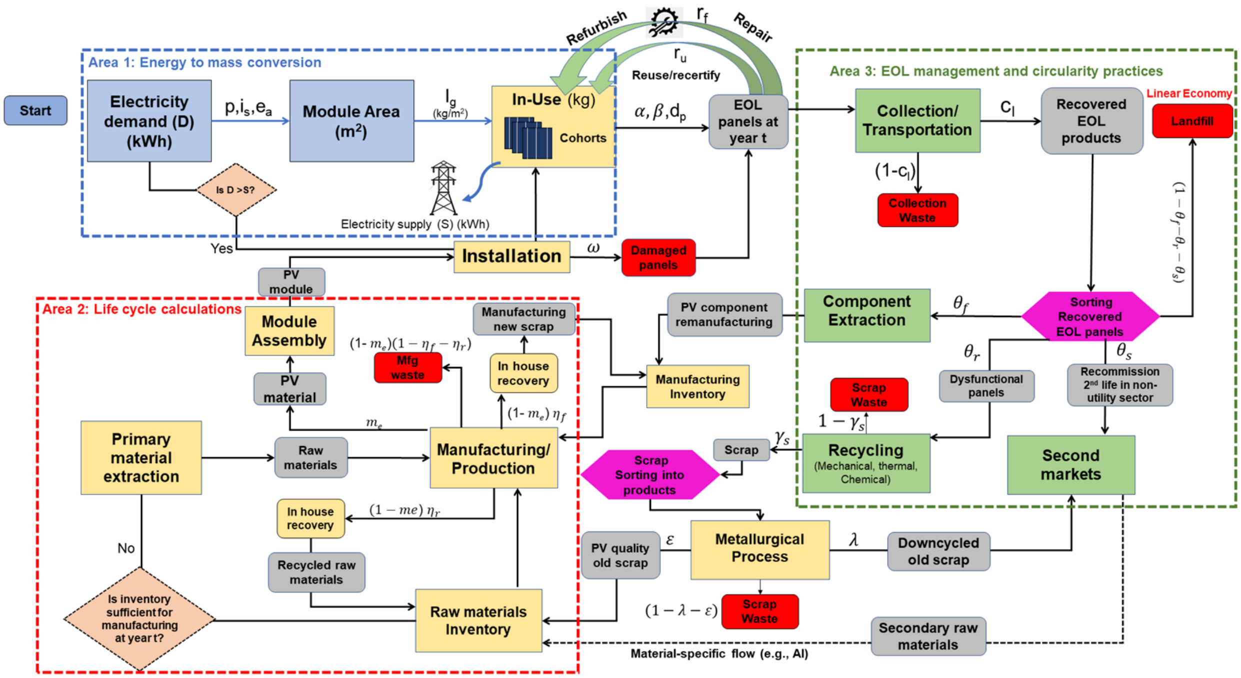


Figure 2: System boundary and parameter definitions of the Photovoltaic Dynamic Material Flow Assessment (PV DMFA) Model.

PV DMFA mines historical utility-scale PV electricity consumption values and predicts U.S. PV electricity demand in the future to estimate PV module area and subsequent material needs through built-in calculation sequence and PV-specific parameters that influence end-use electricity generation. A key difference in the model framework is that the starting point is electricity generation rather than installation capacity. This choice is justified for the following reasons:

1- The need to capture larger scope of PV parameters that have direct impact on installations (e.g. PV deployment location, Performance (derate) ratios).
2- Modeling the impacts of grid operator decisions to manage electricity supply (e.g. curtailment, storage) that may affect PV power plant system sizing and material consumption consequently.
3- The availability of more robust regression tools for future electricity demand predictions that are well-studied and trusted by the research community.

2.2 Framework and System boundary

The model applies a cradle-to-cradle process-based life cycle analysis (LCA) framework to quantify the stocks and flows of PV materials in utility-scale PV markets in the United States based on direct PV electricity available to the grid. Considering direct PV electricity generation allows users to track energy and materials independently and test a wide range of generation parameters (i.e., solar insolation, grid curtailment, derate ratio). Figure 2 illustrates the different areas of PV DMFA model development.

Area 1: Energy-to-Mass conversion

Knowing how much electricity needs to be delivered to the grid is the first step to determine how much material needs to be present in the economy. Below is a stepwise calculation scheme used in the PV DMFA model to track PV electricity generation and losses and to calculate material inflows, stocks, and outflows of the use phase.

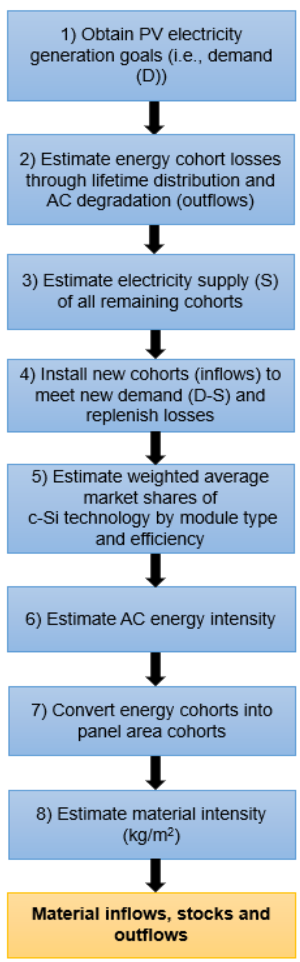


Figure 1. Iterative calculation framework for Area 1: energy to mass

Area 2: Life Cycle Calculations

Conservation of mass is invoked at each node of the control volumes inside the system boundary using pre-defined process yield parameters. Material life cycle stages considered are primary material acquisition (e.g., mining), upstream material processing into PV- quality, in-house manufacturing repurposing, recovery and recycle, module assembly and prospective integration of EOL scrap into different manufacturing life cycle stages.

Area 3: End of Life Waste Management and Circularity Practices

The model considers a wide range of waste management practices that could be part of PV circular economy in the future. Circularity practices could include but not limited to (excluding landfilling): module repairs to prolong system service lifetime, manufacturer recertification for a second life, remanufacturing/repurposing an old module component to make a new module and selling/donating degraded modules to non-utility sectors and finally material extraction through recycling and subsequent metallurgical processing.

2.3 File Structure

Figure 3: File Structure of the code and Excel Input sheet through Juypter Notebook user interface

In order to run the model, it requires an Excel input sheet that feeds the model process parameters. The format of the Excel sheet is discussed in Section 2.2. The Excel file must be in the same directory as the code in order to run as shown in Figure 3.

2.4 Input Data

The Excel file, **PV Input Sheet.xlsx**, must contain the module and material process parameters throughout the years.

Year	D	Beta	Alpha	Beta_2	Alpha_2	iS	p	ru	g_me	g_mre	g_eta_r	g_eta_f	g_cl	g_theta_	g_theta_	g_theta_r	g_gamm	g_epsilo	g_lam	a_me	a_mre	a_eta_r	a_eta_f	a_cl	a_theta_	a_theta_	a_theta_r	a_gamm	a_epsilo	a_lam	Yearly Av	omega	dp1	dp
2000	0.543	25	2.4	25	2.4	1700	0.8	0	0.83	0.95	0.3	0	0	0	0	0	0	0	0	0.25	0.9	0.13	0	0	0	0	0	0	0	0	0.05	0.005	0.02	0.005
2001	5.43E-01	25	2.4	25	2.4	1700	0.8	0	0.83	0.95	0.3	0	0	0	0	0	0	0	0	0.25	0.9	0.13	0	0	0	0	0	0	0	0	0.05	0.005	0.02	0.005
2002	5.55E-01	25	2.4	25	2.4	1700	0.8	0	0.83	0.95	0.3	0	0	0	0	0	0	0	0	0.25	0.9	0.13	0	0	0	0	0	0	0	0	0.05	0.005	0.02	0.005
2003	5.34E-01	25	2.4	25	2.4	1700	0.8	0	0.83	0.95	0.3	0	0	0	0	0	0	0	0	0.25	0.9	0.13	0	0	0	0	0	0	0	0	0.05	0.005	0.02	0.005
2004	5.75E-01	25	2.4	25	2.4	1700	0.8	0	0.83	0.95	0.3	0	0	0	0	0	0	0	0	0.25	0.9	0.13	0	0	0	0	0	0	0	0	0.05	0.005	0.02	0.005

Figure 4: Example Excel Input Sheet

The format of the input sheet is shown above in Figure 4. The first row are the column headers, signifying the process parameters. For each year being studied, a number must be inputted.

Module Process Parameters

Year: int year

Demand [kWh]: energy demand to be fulfilled by PV panels

Beta [years]: Panel's mean expected lifetime

Alpha: Shaped parameter of PV panels

Beta_2 [years]: float panel’s lifetime warranty if panel enters reuse stream

Alpha_2: float Shaped parameter of PV if panel enters reuse stream

iS [kWh/m^2]: float Solar insolation

p [%]: performance ratio of panels

ru [%]: percent of panels that are reused

rf [%]: percent of panels that are refurbished

Yearly Average Curtailment [%]: percent of energy capacity that is curtailed

omega [%]: percent of installation that is lost.

dp [%]: percent of degradation each year

dp1 [%]: percent of degradation in the first year of service

Material Process Parameters

material_me [%]: manufacturing efficiency

material_mre [%]: percent of manufacturing waste that can be recovered

material_eta_r [%]: percentge of recovered manufacturing waste that can be recycled into raw materials

material_eta_f [%]: percentof recovered manufacturing waste that can be remanufactured into refined parts
material_cl [%]: percent of end of life losses that can be collected

material_theta_s [%]: percent of collected eol material that can be sold to secondary markets

material_theta_f [%]:#percentage of collected eol material that can be remanufactured

material_theta_r [%]: percentage of collected eol material that goes to a recycling facility

material_gamma_s [%]: scrap yield of recycling facility

material_epsilon [%]: percentage of scrap that can be turned into PV grade material/PV market penetartion for PV scrap

material_lam [%]: percentage of scrap that can not be turned into PV grade material but can still be reused in a secondary market

maerial_2in_ratio [%] (optional): percent of raw material demand that is met with imported material

Input data sources:
To be able to run the model successfully, user must provide the appropriate input data. For the baseline scenario, authors leveraged roadmap market reports, literature, and expert opinions mostly at NREL. A detailed citation list for different input parameters is available from the peer reviewed article expected to be published along this code. The following is a truncated list of all main sources available in the model input sheet for the baseline scenario:

Electricity generation: historical data is mined from EIA AEO reports; future data is collected from GCAM model as a courtesy from Gokul Iyer at PNNL. A logistic fit is used to fit historical and projected data to obtain a smoother and more statistically significant curve.EIA data: U.S. Energy Information Administration (EIA), https://www.eia.gov/todayinenergy/detail.php?id=34112, (accessed 13 January 2021).

PV technology type, module efficiencies and embedded material intensities: International Technology Roadmap for Photovoltaic Reports (ITRPV) editions (2010-2021) were leveraged for historical and projected PV parameters. Please refer to source code for additional details about average weighted market share calculations and assumptions about technology market shares.

PV waste model: According to PV DMFA model, PV modules retire if the cumulative probability lifetime losses reach 98% OR if the cumulative efficiency degradation (i.e., retirement efficiency) reaches 80% of that at installation. A 2-parameter Weibull probability distribution is used to model the random module lifetime losses. Weibull parameters (alpha and beta) were tested in a variety of scenarios. The baseline scenario assumes an early loss scenario Weibull parameters as depicted by IRENA 2016 report by IEA/PVPS.

Material processing: Case specific parameters are calculated from Ullmann's chemical encyclopedia for flat glass, aluminum 6063 alloys, PV grade silicon and silver. All parameters were obtained by material balance datasheets for the respective material production process. Please refer to the Supporting information document of the journal articles published for this work.

PV recycling lines, scrap yield and scrap quality allocation: The exit stream from sorting step that goes into PV recycling indicates material extraction through module disassembly. Thus far, PV recycling markets are niche. Authors relied on the few established lines in Europe and Japan (i.e., Veolia, FRELP and NEDO FAIS). Scrap quality is of major concern for the technical feasibility of closed loop recycling in PV. In PV DMFA, scrap quality is modelled by tracking the bulk components in materials which in itself indicative of the material purity. For example, Aluminum 6063 alloy is at most 97.5% aluminum, Mg 0.45-0.9%, Fe Max 0.35%, etc. While PV DMFA model does not track trace alloys, it does track bulk Al. So if aluminum purity slips below 97.5%, this stream cannot be assumed to offset any value in manufacturing needs. However, it might be assigned to secondary markets. It is important to emphasize that scrap purity allocation may not be readily applicable to some non-alloy materials like flat glass since material composition seldom changes. Instead, neighboring material contamination could occur on a process-specific basis. Authors relied on recycling industry feedback to make educated assumptions for the allocation of these non-alloy materials based on purity and nature of recycling process studied. Please see more details in the source code.

2.5 Output Data

The model calculates the virgin material, recycled material, and secondary markets impact for each year. The results are added in their own arrays and can be printed out to the console if needed. The following are the key arrays used:

Energy Array Outputs

yearly_capacity [kWh]: summation of all the cohorts' installed energy capacity

yearly_capacity_loss [kWh]: summation of all the cohorts' energy waste from installation, weibull, and degradation

yearly_installed: yearly capacity installed, which is the diagonal of the cohort capacity arrays.

Material Array Outputs

material_losses [kg]: summation of all glass installation losses

material_losses_eol [kg]: sum of the material exiting the use stream

material_i_losses [kg]: material installation losses

material_in_use [kg]: total amount of glass in use from all cohorts

material_installation [kg]: amount of material installed

material_cumu_installation [kg]: cumulative material installed

material_eol [kg]: cumulative sum of g_losses_eol

Manufacturing Material Array Outputs

material_direct_mfg [kg]: amount of material needed

material_resource_extraction [kg]: amount of raw material needed to be manufactured

material_yearly_waste [kg]: material that cannot be recycled or collected

material_mfg_inventory [kg]: the current stock of manufactured material that is not in service

material_material_inventory [kg]: the current stock of raw materials that have been extracted but not yet manufactured

material_re_man_inventory [kg]: the current stock of material that can be remanufactured in another application material_second_market [kg]: remanufactured material destined for second market applications

material_mfg_loss_sum [kg]: cumulative sum of manufacturing losses

material_mfg_recycle_sum [kg]: cumulative sum of recovered material that can be recycled

material_eol_collection_sum [kg]: cumulative sum of material that can be recaptured

material_eol_collection_loss_sum [kg]: cumulative sum of collection losses

material_scrap_sum [kg]: cumulative sum of scrap that is produced from the recycled material

material_scrap_waste_sum [kg]: cumulative sum of scrap losses

material_scrap_recycle_sum [kg]: cumulative sum of scrap that is remanufactured into PV grade material

material_mfg_waste_sum [kg]: cumulative sum of manufacturing waste

material_mfg_re_man_sum [kg]: cumulative sum of remanufactured material that was recovered

material_scrap_second_market_sum [kg]: cumulative sum of scrap that is remanufactured into PV grade material but enters a secondary market

material_eol_losses_sum [kg]: cumulative sum of material that cannot be re-purposed

material_mfg_recovery_sum [kg]: cumulative sum of manufactured losses that can be recovered

material_mfg_unrecovered_sum [kg]: cumulative sum of unrecoverable manufactured losses

material_cumu_waste [kg]: cumulative sum of all material waste

material_cumu_extract [kg]: cumulative sum of raw extraction of material

material_eol_reman_sum [kg]: cumulative sum of collected material that can be remanufactured into PV components

material_PV_second_market_sum [kg]: cumulative sum of collected material that can be sold to a secondary market

3 Simulation

This section outlines running the model through Juypter Notebook step by step in which the code is ordered by.

3.1 Energy and Area Calculations

The model is fairly simple to run using Juypter Notebook. Launch Juypter Notebook via Anaconda Navigator. Open the file, **DMFAforPV.ipynb**, and make sure your Excel file is setup and in the same directory as the code as explained in Section 2.3. This section will go through each section of the code and briefly explain its purpose.

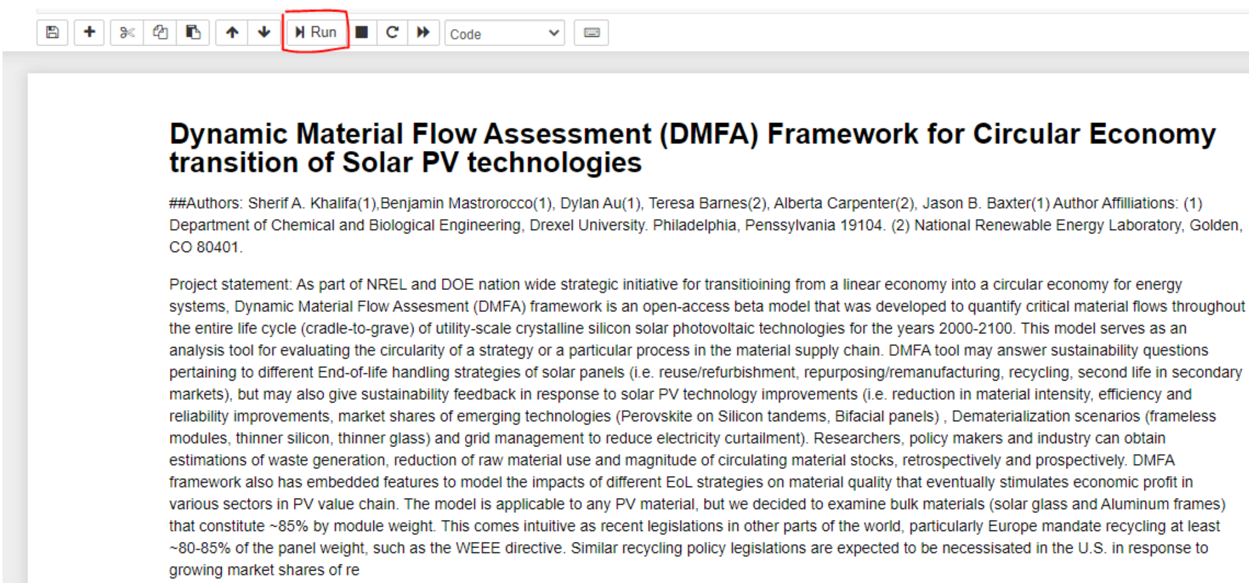


Figure 5: Juypter Notebook user interface

As seen in Figure 5, the **Run** button can be hit and the highlighted section of code will run. The Run button should be hit until the end of the code is reached. In Figure 5, the block of code outlines the project.

Required packages

```
In [2]: import sys
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        from openpyxl import load_workbook
        import plotly.offline as pyo
        import plotly.graph_objs as go
        pyo.init_notebook_mode()     # Set notebook mode to work in offline

        pd.set_option('display.max_columns', None)   #these lines improve the visibility of the array
        np.set_printoptions(threshold=sys.maxsize)
        np.set_printoptions(edgeitems=50, linewidth=1000000,
            formatter=dict(float=lambda x: "%.3g" % x))

        np.set_printoptions(edgeitems=10)
        np.core.arrayprint._line_width = 180
```

Figure 6: Required installed packages as explained in Section 2.2

In Figure 6, shows the next section of code that uses the installed packages mentioned in Section 1.2.

Extracting data from Excel input sheet

```
In [3]: #User must create/download an Excel input sheet and set the Excel in the same directory as the code

        input_sheet = pd.read_excel("PV Input Sheet V18.xlsx")

        beta  = list(input_sheet['Beta'])  #Panel's mean expected lifetime (year)
        alpha = list(input_sheet['Alpha']) #shaped parameter for two-shaped parameter weibull distribution
        c_beta = list(input_sheet['Beta_2']) #Panel's lifetime if it is reused (year)
        c_alpha = list(input_sheet['Alpha_2']) #Shaped parameter for the weibull distribution if it is reused
        D = list(input_sheet['D']) #Yearly energy demand (kWh)
        iS = list(input_sheet['iS']) #Yearly solar insolation (kWh/m2)
        p = list(input_sheet['p']) #performance ratio of the panel
        ru = list(input_sheet['ru'])  #Percentage of panels that are reused
        rf = list(input_sheet['rf'])     #ratio of panels in the reuse loop that can be refurbished
        omega = list(input_sheet['omega']) #Percentage of panels lost during instalation and transportation

        #variables using "g_" represent parameters for glass variables using "a_" represent parameters for aluminum

        g_me = list(input_sheet['g_me'])  #manufacturing efficiency
        g_mre = list(input_sheet['g_mre'])  #percentage of manufacturing waste that can be recovered
```

Figure 7: Code that uses the Excel Input sheet to get process parameters

In Figure 7, the section of code extracts data from the Excel Input Sheet, PV Input Sheet.xlsx. It uses the first row of the Excel sheet to signify the process parameter for the code. See Section 2.2 for the variable names for Excel Input Sheet.

Average Panel Efficiency Calculation

```
#This section calculates yearly average panel efficiencies

#efficiency calculation 2000-2017 Source:SPV 2018-2019
market_mono = np.array([0,0,0,0,0,0,0,0,0,0.35,0.35,0.37,0.38,0.37,0.36,0.39,0.35,0.41,0.45])
market_ntype = np.array([0,0,0,0,0,0,0,0,0,0,0,0.18,0.18,0.18,0.18,0.18,0.18,0.18,0.18])
market_multi = np.array([1,1,1,1,1,1,1,1,1,0.49,0.46,0.48,0.48,0.52,0.55,0.55,0.58,0.54,0.51])

efficiency_mono = np.array([0.085,0.09,0.096,0.101,0.106,0.111,0.116,0.122,0.127,0.1345,0.137,0.141,0.1479,0.151,0.1585,0.165,0.1
efficiency_ntype = np.array([0.118,0.124,0.13,0.136,0.141,0.147,0.153,0.159,0.165,0.1745,0.1792,0.1804,0.1845,0.192,0.1989,0.202,
efficiency_multi = np.array([0.088,0.093,0.097,0.101,0.106,0.11,0.114,0.119,0.123,0.1301,0.132,0.137,0.1386,0.145,0.149,0.151,0.1

ea1 = (((market_mono - market_ntype) * efficiency_mono) + (market_ntype * efficiency_ntype) + (market_multi * efficiency_multi))/

ea = []
```

Figure 8: Code that calculates average panel efficiency

In Figure 8, the section of code calculates the average panel efficiency each year based on various sources, including various cell technology that is used.

Functions:

```
#this function takes a given b value and generates a list containing every year in the cohort's lifetime
def life( final_year ):
    life = np.arange( 1, final_year + 1 )
    lifetime = final_year
    return life

#this function returns the cumulative weibull probability of failure for a given cohort at a given point in its lifetime
def weibull_c( beta, alpha, age ):
    cumu = 1 - np.exp( -( age / beta ) ** alpha )
    return cumu

#calculates cumulative dagradation of a cohort for a given year
def degradation(dp,dp1,age):
    if age == 1:
        deg = dp1
    else:
        deg = dp1 + dp * (age-1)
    return deg

#calculates energy intensity(AC electricity) from solar insolation, efficiency, and performance
def calcAC(years):
    ac_electricity = []
    for year in years:
        if year <= final_year:
            acelec = p[ year - 1 ] * iS[ year - 1 ] * ea[ year - 1 ]
            ac_electricity.append( acelec )
    return ac_electricity
```

Figure 9: Section of code that encapsulates various functions used for calculating energy and area flows

In Figure 9, the functions are calculations used in the energy and area flow calculations. These functions work by accepting a set of parameters needed to fulfill the calculation. For example, in order to calculate the cumulative two-shaped Weibull distribution, beta, alpha, and the age of the panel must be passed through in order for the calculation to be done.

Cohorts' Energy and Area Calculation:

```
6]:
f_loop = [ ]
c_loop = [ ]

#In these arrays, each row represents a cohort, and each column represents a year
#12 arrays are being calculated here. The first four represent Energy In use, Energy Loss, Area In Use, and Area Loss in the fi
#The other 8 represent the two second life cycle options, refurbishment and recertificaiton

for year in years:  #starts year 1
    if year == 1:
        S = 0   #it is assumed that there was 0 solar energy generation before the year 2000
    else:
        S = calcSupply(year,total_in_use,c_total_in_use,f_total_in_use)     #calcualtes energy suply each year
```

Figure 10: Section of code that calculates each cohort energy and area flows

In Figure 10, the block of code shown is the cohorts' energy and area calculation. This loops through the fiscal years put in the Excel Input sheet and calculates energy and area capacity, installation, and losses. These utilizes the functions in the Functions section of code as seen in Figure 9.

Yearly energy and area in use, installation, and waste

```
In [6]: #Yearly energy and area arrays
yearly_installed = total_in_use.diagonal( )     #the diagonal of the total_in_use array represents the yearly capacity installed
yearly_capacity_loss = np.sum( total_waste, axis = 0 ) + np.sum( c_total_waste, axis = 0 ) + np.sum( f_total_waste, axis = 0 )  #
yearly_capacity = np.sum( total_in_use, axis = 0 ) + np.sum( c_total_in_use, axis = 0 ) + np.sum( f_total_in_use, axis = 0 )
m2_yearly_installed = m2_total_in_use.diagonal( )     #the diagonal of the m2_total_in_use array represents the anual area of ir
yearly_m2_loss = np.sum( m2_total_waste, axis = 0 ) + np.sum( c_m2_total_waste, axis = 0 ) + np.sum( f_m2_total_waste, axis = 0 )
yearly_m2 = np.sum( m2_total_in_use, axis = 0 ) + np.sum( c_m2_total_in_use, axis = 0 ) + np.sum( f_m2_total_in_use, axis = 0 )

#Displays arrays
```

Figure 11: Section of code that calculates yearly energy and area values from cohort values

In Figure 11, shows the yearly energy and array values are calculated. Yearly installation values are the diagonal of the cohort's capacity array. Yearly energy and area capacity and waste are the sum of each column of the cohort arrays.

3.2 Material Calculations

Next are the material calculations done in the model. The model has currently calculated glass and aluminum mass flows but only glass will be shown here as an example. Other materials of interest can be done as long as enough information is provided.

Glass Material Intensity Calculation

```
In [7]: #This section calculates the amount of glass per area of panels.

t2 = []  #average thickness of glass per module (mm)
t1 = []  #average thickness of glass (mm)

x1 =   [0.96,0.93, 0.90, 0.88, 0.86, 0.83, 0.80, 0.78, 0.75, 0.72, 0.697, 0.674, 0.65]     #market share of 3mm thick glass shee
x2 = [0.04,0.07, 0.10, 0.12, 0.14, 0.16, 0.18, 0.20, 0.22, 0.24, 0.26, 0.28, 0.30]     #market share of 2.5mm thick glass sheets
x3 = [0.00,0.00, 0.00, 0.00, 0.00, 0.01, 0.02, 0.02, 0.03, 0.04, 0.043, 0.046, 0.05]     #market share of 1.8mm thick glass shee

biratio = [0,0,0,0,0,0,0,0,0,0,0,0,0.005,0.01,0.015,0.02,0.02,0.2,0.24,0.28,0.32,0.35,0.39,0.4,0.4,0.42,0.44,0.47,0.5]     #Mark

for year in years:
```

Figure 12: Material Intensity calculations that is used to convert area to mass of material

In Figure 12 is the material intensity calculation specifically for glass. The material intensity is defined as the mass of material per area, which is obtained from literature values.

Glass Material Cohort and Yearly Calculations: ¶

```
In [8]: #Material cohort arrays
for year in years:     #converts area values to kg of glass values
    glass_in_cohort = m2_total_in_use[ ( year - 1 ) : year ] * g_Ig[ year - 1 ] + c_m2_total_in_use[ ( year - 1 ) : year ] * g_Ig
    if year == 1:
        total_glass = np.array( glass_in_cohort )
    else:
        total_glass = np.append( total_glass, glass_in_cohort, axis = 0 )

for year in years:     #calculates glass waste values
    glass_cohort_waste = m2_total_waste[ ( year - 1 ) : year ] * g_Ig[ year - 1 ] + c_m2_total_waste[ ( year - 1 ) : year ] * g_I
    if year == 1:
        glass_waste = np.array( glass_cohort_waste )
    else:
        glass_waste = np.append( glass_waste, glass_cohort_waste, axis = 0 )



#yearly material arrays
g_losses = np.sum(glass_waste,axis=0) - glass_waste.diagonal()     #install losses are counted in a diferent stream
g_losses_eol=np.sum(glass_waste,axis=0) #Calculates total amount of panels exiting use phase
g_i_losses = glass_waste.diagonal()     #install losses are represented by the diagonal of the waste array
g_in_use = np.sum(total_glass, axis=0)     #the sum of each column in the total_glass array represents the ammount of glass in us
g_installation = np.diagonal(total_glass)     #the diagonal of the total_glass array represents the total mass of glass installed
g_cumu_install=np.cumsum(g_installation)     # sums EoL panels to obtain cumulative sum
g_eol=np.cumsum(g_losses_eol)     # sums EoL panels to obtain cumulative sum
```

Figure 13: Material cohort and yearly calculations

In Figure 13, the material cohort and yearly values are calculated by converting the values using the respective material intensity value.

Glass Manufacture/Recycle Calculations

```
In [9]: #Manufacturing arrays
g_direct_mfg = [ ]     #kg of glass that is required to be manufactured each year
g_resource_extraction = [ ]     #kg of raw material required to supply the manufacturing needs
g_yearly_waste = [ ]     #kg of waste that can not be recycled or collected
g_mfg_inventory = [ ]     #the current stock of manufactured glass that isnt in service
g_material_inventory = [ ]     #the current stock of raw materials that have been extracted but not yet manufactured
g_re_man_inventory = [ ]     #the current stock of material that can be remanufactured in another application
g_second_market = [ ]     #remanufactured glass destined for second market applications


g_mfg_loss_sum = []     #the 'sum' variables are used to track material flows for the Sankey Diagram
g_mfg_recycle_sum = []
g_eol_collection_sum = []
g_eol_collection_loss_sum = []
```

Figure 14: Material manufacturing mass flows

In Figure 14 highlights the mass flows with manufacturing, specifically for glass. These variables are listed in Section 2.2. These calculations are based on the literature values given in the Excel Input sheet and are mostly split fractions that determines how much mass goes to each flow.

Glass Mass Balance Error Calculations

```
In [10]: g_error = []

for year in years:
    g_i = sum(g_resource_extraction[0:year])     #sum of all material that has entered the system
    g_o = sum(g_yearly_waste[0:year]) + sum(g_second_market[0:year]) + sum(g_re_man_inventory[0:year])     #sum of all material t
    g_a = g_in_use[year-1] + g_inventory[year-1]     #ammount of material currently in the system
    g_error.append((((g_i-g_o)-g_a)/g_in_use[year-1])*100)     #calculating overall mass balance and percent errror

plt.plot(timescale,g_error)
plt.title('Glass Material Balance Error')
plt.ylabel("% error")
plt.show()

#this "error" should never surpass 1*10^-10 % and should also not have a distinct trend
```

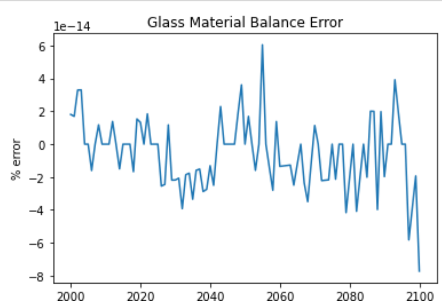


Figure 15: Yearly material balance calculations

In Figure 15, a material balance, specifically for glass, is done on the system to ensure that all mass is accounted for. The values should be 0 as shown in the plot.

Sankey Glass Diagram Generator

```
#This code generates a sankey diagram of the DMFA
#The nodes(labeled boxes) can be dragged around to better visualize the flows
#Mousing over the flows can reveal more information
#This sankey diagram can represent a single year or a sum over many years
#The diagrma can be downloaded as a png by mousing over the oper right corner of the diagram and selecting the camera icon
#!!!! This Code May Not Work In Jupyter Lab, Jupyter Notebook is Recomended !!!!
```

Figure 16: Section of code that generates a Sankey diagram

In Figure 16 is the section of code that generates a Sankey diagram. It plots various flows such as resource extraction, end of life collection, manufacturing waste, etc.

The above figures represent the code and the order of the code in its current state.

3.3 Results

This section will outline the results generated from the model when a user runs through the blocks of code in Section 3.1 and 3.2. The results include generated plots for users to look at by default. If the user wants to see a specific variable and its value, you can use the built in print function from Python to print the results for you to see. If the user wants to see a specific plot of different variables that are not shown in the following section, please refer to matplotlib documentation to learn how to plot variables in Python.

The following results are based on the baseline scenario input as seen in the PV Input Sheet.xlsx.

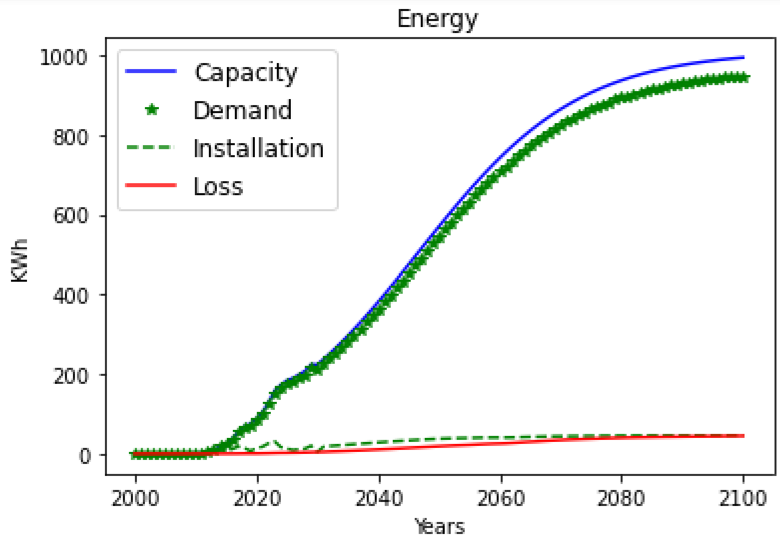


Figure 17: Plot of various energy flows vs. time

In Figure 17, the energy capacity, demand, installation, and loss are plotted. This figure is plotted to ensure that the energy demand is being met and that more energy is being installed to account for losses.

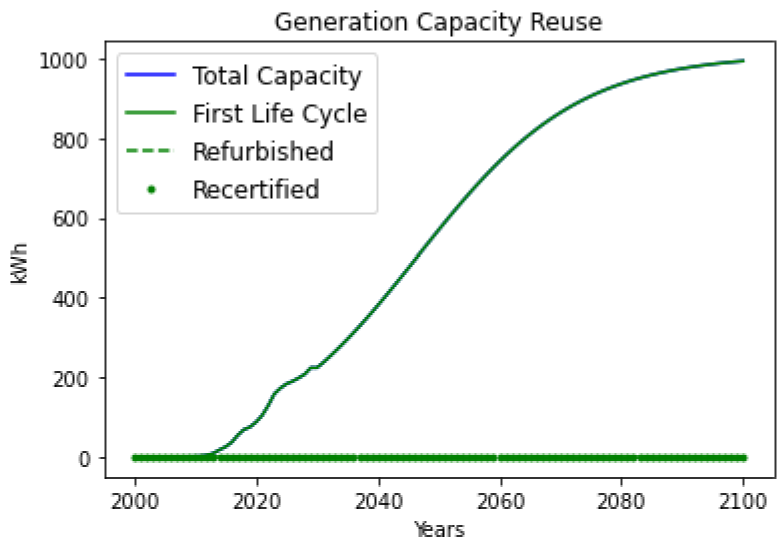


Figure 18: Energy capacity from panels in different lifecycles

In Figure 18, panels in its first lifecycle, panels that have been refurbished, and panels that have been recertified are plotted to see the impact of reusing panels.

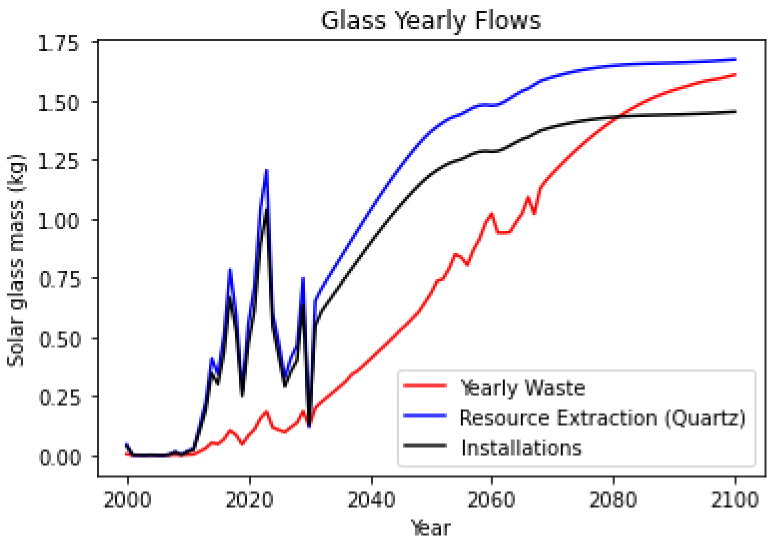


Figure 19: Yearly material flows for glass

In Figure 19, resource extraction, installations, and waste are plotted to see how much material is needed, how much is actually installed, and how much goes to waste.

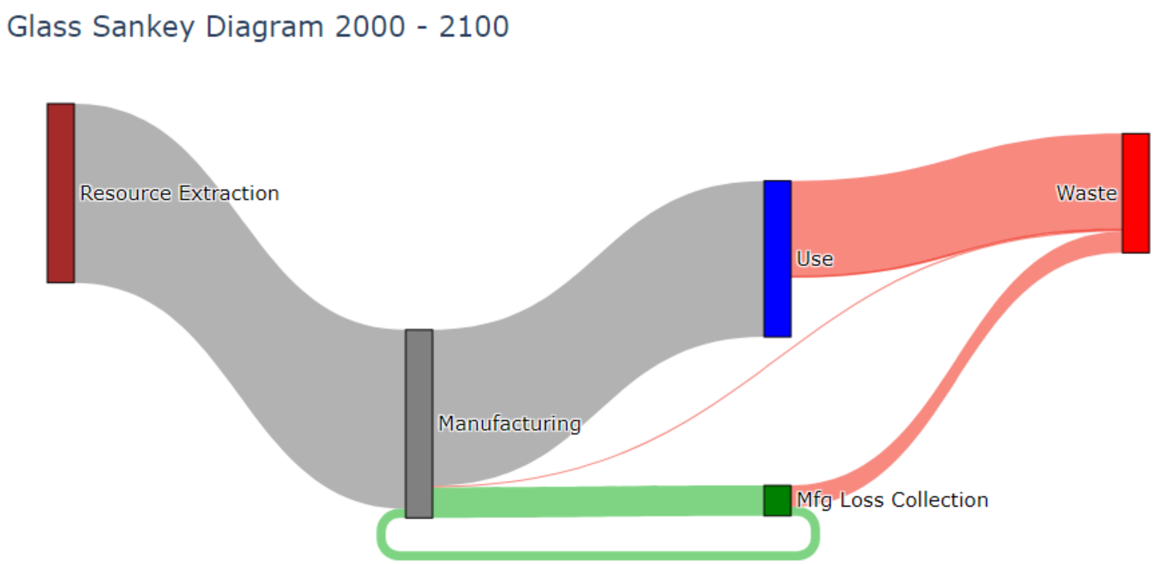


Figure 20: Glass Sankey Diagram

In Figure 20 is a Sankey diagram specifically for Glass. The Sankey diagram helps visualize how much of the material goes to each stream and to determine which stream contributes the most.

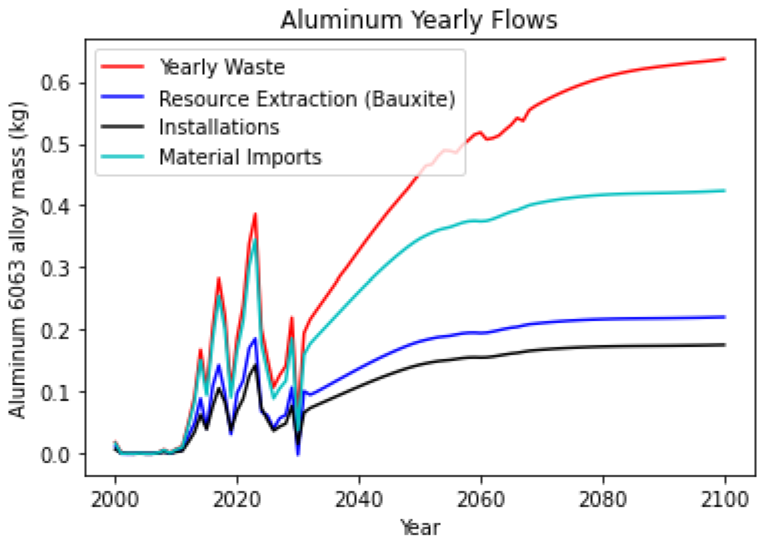


Figure 21: Aluminum Mass Flows

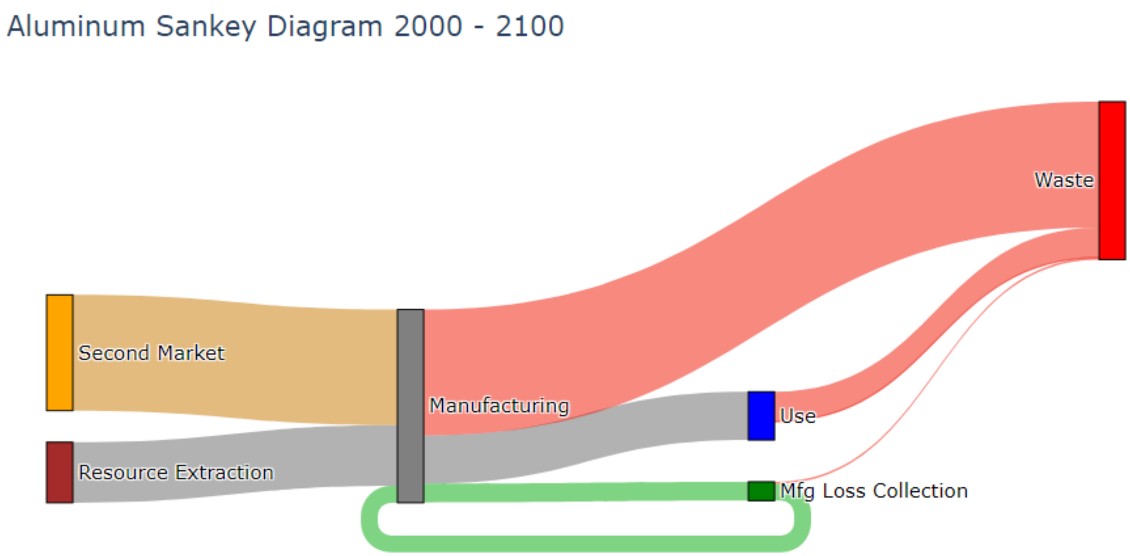


Figure 22: Aluminum Sankey Diagram

In Figure 21 and 22 are the same plot outputs but for Aluminum. As long as the user implements and gives a material intensity, the user can trace any material and get the same output.

```
Cumulative Glass Waste: 73.39 (Metric Tons)
Cumulative Al Waste: 38.70 (Metric Tons)
Cumulative Overall Waste 112.09 (Metric Tons):
Resource Extraction 91.71 (Metric Tons):
```

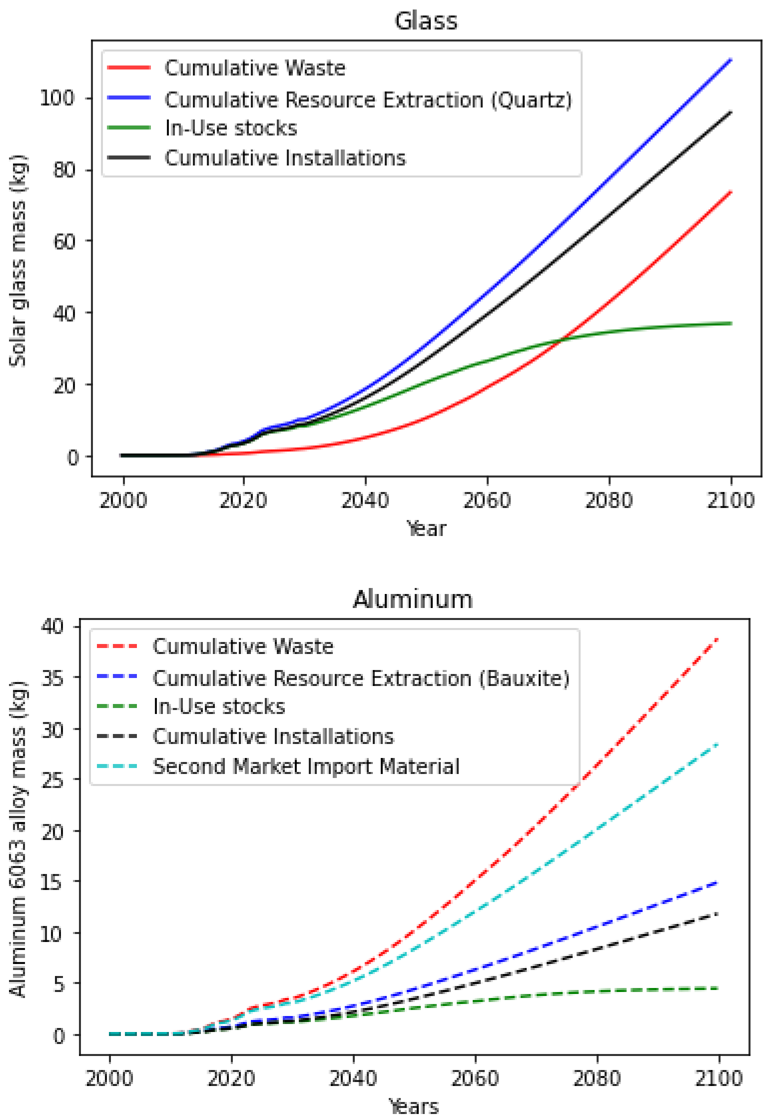


Figure 23: Cumulative material plots

In Figure 23, the cumulative values of the material flows are printed and plotted against time to see how they change for various scenarios.

Finally, the cohort arrays are outputted to an Excel file called **results.xlsx.** The cohort arrays are exported since the table can be quite big for longer years. Therefore, for a user friendly experience, it will be easier to see those values through Excel.

Cohort Arrays Outputted to Excel

[34]:
```
#Represent the 12 cohort arrays for energy and array
df1 =pd.DataFrame(total_in_use)
df2 =pd.DataFrame(c_total_in_use)
df3 =pd.DataFrame(f_total_in_use)

df4 = pd.DataFrame(m2_total_in_use)
df5 = pd.DataFrame(c_m2_total_in_use)
df6 = pd.DataFrame(f_m2_total_in_use)

df7 = pd.DataFrame(total_waste)
df8 = pd.DataFrame(c_total_waste)
df9 = pd.DataFrame(f_total_waste)

df10 = pd.DataFrame(m2_total_waste)
df11 = pd.DataFrame(c_m2_total_waste)
df12 = pd.DataFrame(f_m2_total_waste)

df13 = pd.DataFrame(total_glass)
df14 = pd.DataFrame(glass_waste)
df15 = pd.DataFrame(total_al)
df16 = pd.DataFrame(al_waste)
```

Figure 24: Exporting cohort arrays to Excel

In Figure 24 is the section of code that exports the cohort arrays to Excel. This is **optional** to run. If a user wants to export other arrays to Excel, follow the format as seen in the section of code and it can be done. This can be used for testing purposes if someone needs to extract values for other uses.