

Proyecto de Investigación

Cristhiam Daniel Campos Julca

25 de agosto de 2022

1. Simulación en Matlab

En primer lugar se implementará en Simulink el modelo de un arreglo fotovoltaico, con paneles del modelo Sunset PX 72, que cuenta con 72 celdas de silicio polycrystalino. Con la finalidad de obtener una potencia aproximada de 4.073 kW se colocan cuatro paneles en serie y tres en paralelo.

En la siguiente tabla se pueden observar los datos proporcionados por el panel Sunset PX a condiciones estándar de prueba, bajo una temperatura de 298.15 K equivalente a 25° C y una radiación de 1000 W/m².

Datos bajo condiciones estándar	STD
Potencia en el punto máximo (P_{max})	340 W
Tensión en circuito abierto (V_{oc})	47.4 V
Tension en el punto de maxima potencia (V_{mpp})	38.4 V
Corriente de cortocircuito (I_{sc})	9.35 A
Corriente en el punto de máxima potencia (I_{mpp})	8.84 A
Numero de celdas (N_s)	72
Coeficiente de Temperatura (I_{sc})	0.037 % /K
Coeficiente de Temperatura (V_{oc})	-0.32 % /K
Resistencia en serie (R_s)	0.39 Ω
Resistencia en paralelo (R_{sh})	545.82 Ω

Para esta primera simulación se considera las entradas constantes, es decir una temperatura de 25°C y una irradiancia de 1000 W/m^2

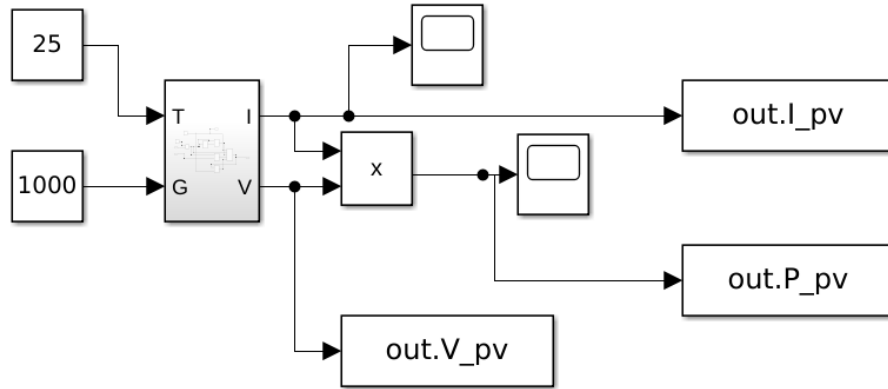


Figura 1: Arreglo PV

Así mismo, se parte del modelo de los cinco parámetros, en donde la corriente y la tensión de salida se ven gobernadas por los siguientes ecuaciones representadas en subsistemas:

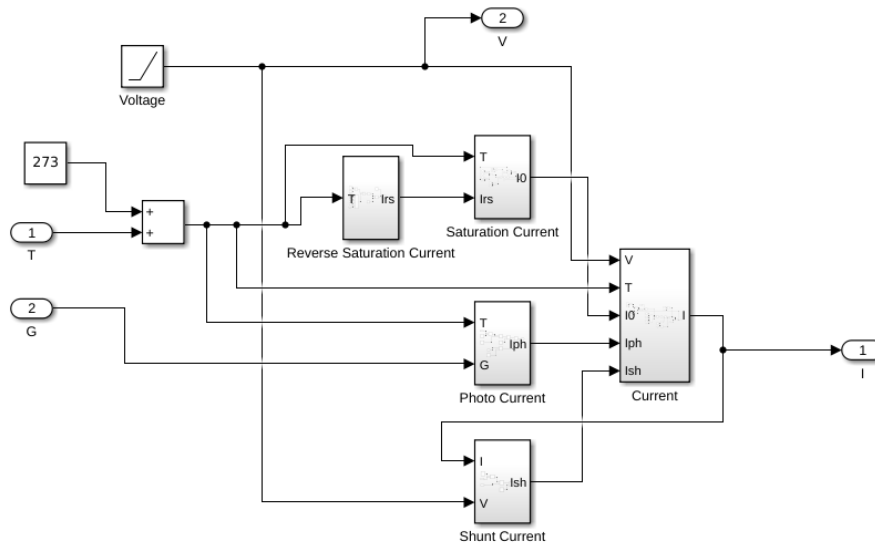


Figura 2: Sub-sistemas del modelo

- Fotocorriente

$$I_{ph} = (I_{sc} + K_i \times (T - 298)) \times (G/1000)$$

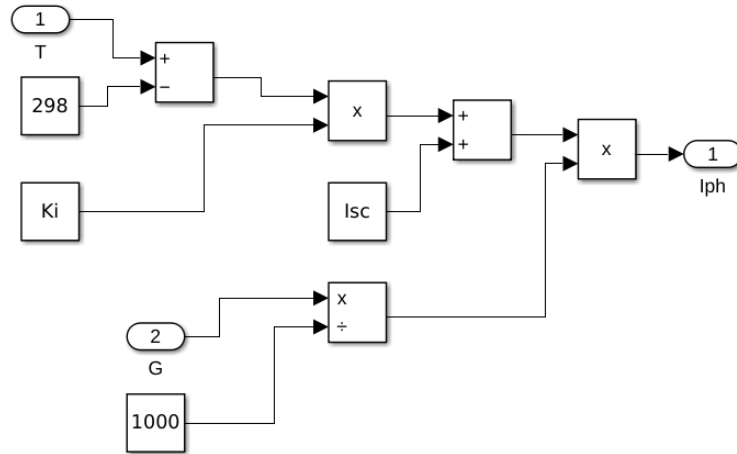


Figura 3: Subsistema para Fotocorriente

- Corriente de saturación

$$I_o = I_{rs} \times \left(\frac{T}{T_n}\right)^3 \times \exp((q \times E_{go} \times (1/T_n - 1/T)) / (n \times k))$$

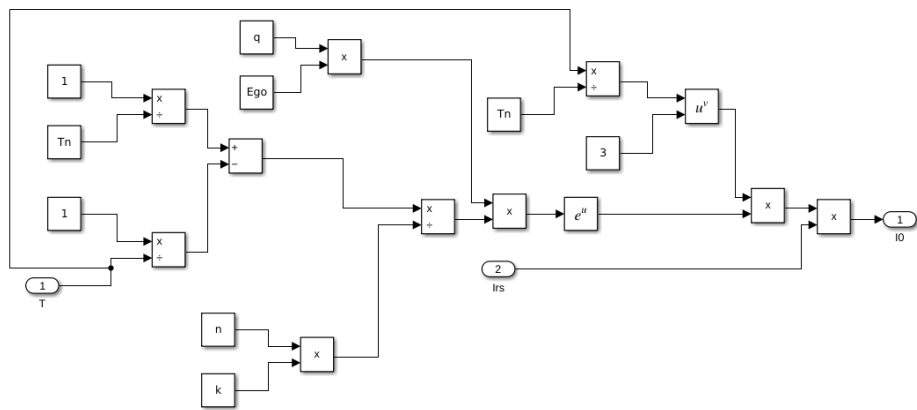


Figura 4: Subsistema de Corriente de saturación

- Corriente de saturación reversa

$$I_{rs} = I_{sc} / (\exp((q * V_{oc}) / (n * N_s * k * T)) - 1)$$

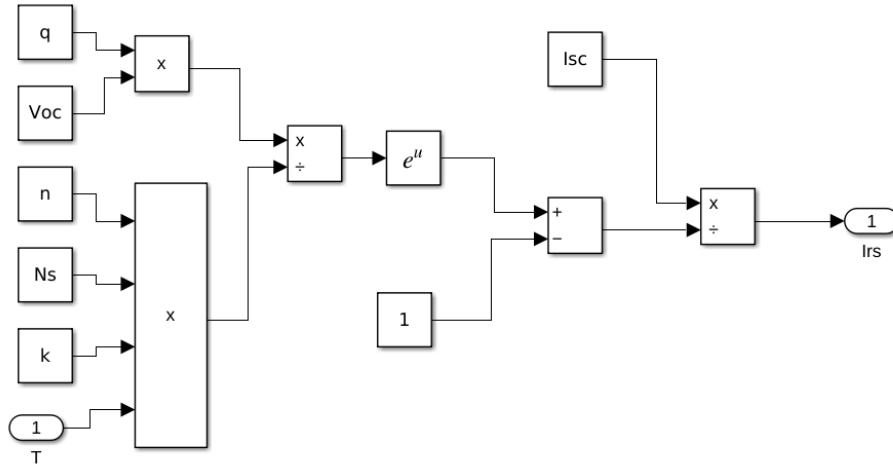


Figura 5: Subsistema de Corriente de saturación reversa

- Corriente shunt

$$I_{sh} = (V + I * R_s) / R_{sh}$$

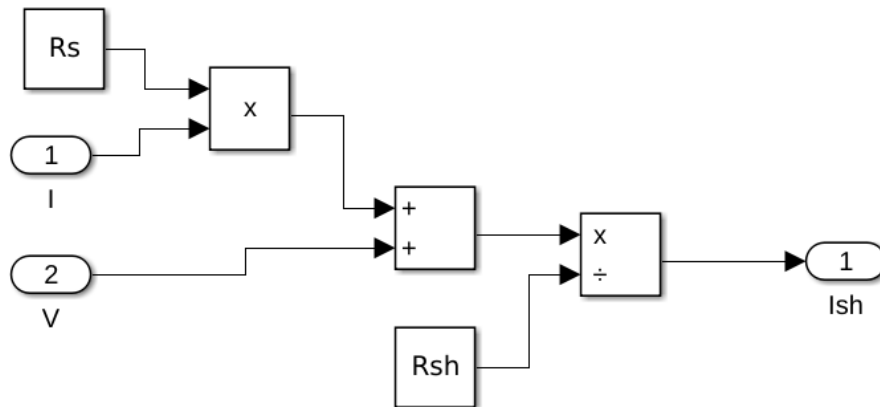


Figura 6: Subsistema de la corriente shunt

- Corriente de salida

$$I = I_{ph} * NP - I_o * NP * (\exp((q * (V / NS + I * R_s / NP)) / (n * N_s * k * T)) - 1) - I_{sh}$$

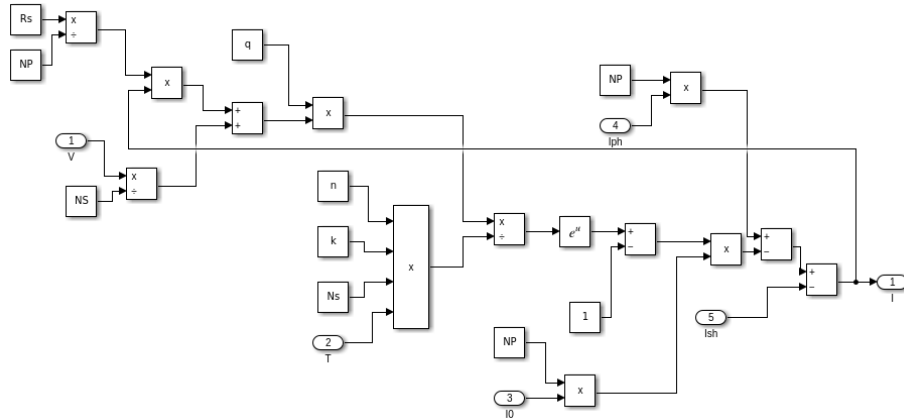


Figura 7: Subsistema de la corriente de salida

Los resultados son almacenados en un archivo CSV para su fácil manipulación en otros lenguajes como Python.

Las curvas características que se obtienen del arreglo fotovoltaico son:

```

1 def curvaCaracteristica():
2     v = df.V_pv
3     i = df.I_pv
4     p = df.P_pv
5
6     fig, ax1 = plt.subplots()
7
8     color = "tab:red"
9     ax1.set_xlabel("Tension (V)")
10    ax1.set_ylabel("Corriente (A)", color=color)
11    ax1.plot(v, i, color=color)
12    ax1.tick_params(axis="y", labelcolor=color)
13    ax1.set_xlim(0,195)
14    ax1.set_ylim(0,30)
15    ax2 = ax1.twinx()
16
17    color = "tab:blue"
18    ax2.set_ylabel("Potencia (W)", color=color)
19    ax2.plot(v, p, color=color)
20    ax2.tick_params(axis="y", labelcolor=color)
21    ax2.set_xlim(0,195)

```

```

22     ax2.set_ylim(0,4000)
23
24     fig.tight_layout()
25     plt.grid()
26
27     plt.savefig("./imagenes/curvacaracteristica.png")
28     plt.show()

```

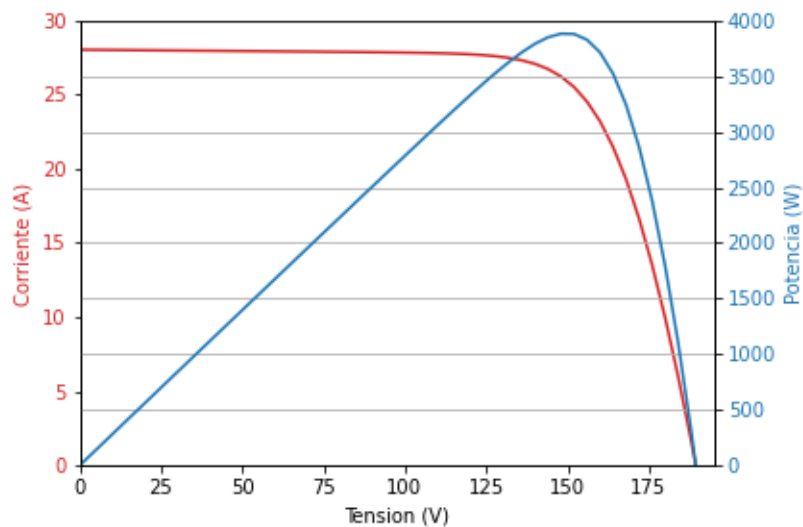


Figura 8: Curvas características

Se hace uso de la función `numpy.poly1d()` que nos ayuda a definir una función polinomial de la siguiente manera:

```

1 def curvasAjuste():
2     #fit polynomial models up to degree 5
3     model1 = np.poly1d(np.polyfit(df.V_pv, df.I_pv, 1))
4     model2 = np.poly1d(np.polyfit(df.V_pv, df.I_pv, 2))
5     model3 = np.poly1d(np.polyfit(df.V_pv, df.I_pv, 3))
6     model4 = np.poly1d(np.polyfit(df.V_pv, df.I_pv, 4))
7     model5 = np.poly1d(np.polyfit(df.V_pv, df.I_pv, 5))
8
9     #create scatterplot
10    polyline = np.linspace(1, 200, 100)
11    plt.scatter(df.V_pv, df.I_pv)
12
13    #add fitted polynomial lines to scatterplot
14    plt.plot(polyline, model1(polyline), color='green',
15            label= 'Funcion grado 1')
16    plt.plot(polyline, model2(polyline), color='red',
17            label= 'Funcion grado 2')

```

```

16 plt.plot(polyline, model3(polyline), color='purple',
17 label= 'Funcion grado 3')
18 plt.plot(polyline, model4(polyline), color='blue',
19 label= 'Funcion grado 4')
20 plt.plot(polyline, model5(polyline), color='orange',
21 label= 'Funcion grado 5')
22
23 plt.legend()
24 plt.grid()
25 plt.savefig("./imagenes/curvaajuste.png")
26 plt.show()

```

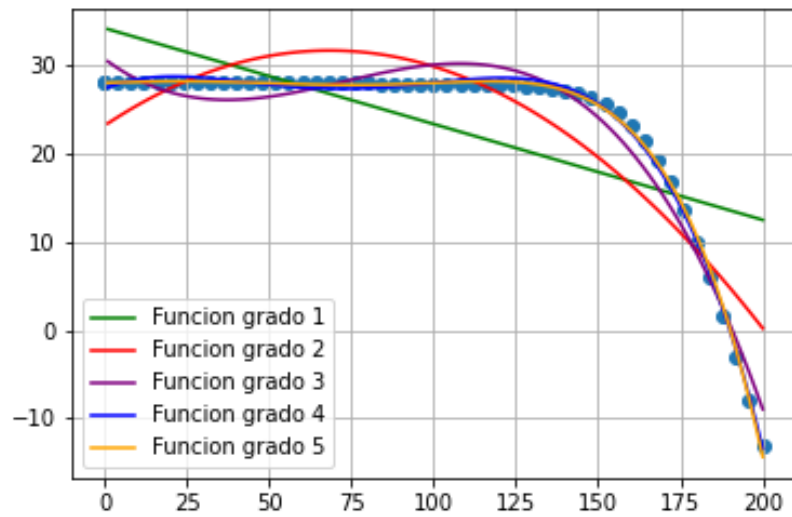


Figura 9: Funciones Polinomiales