Model training for aircraft requirement predictions

Jorge L. Lovaco PhD Candidate

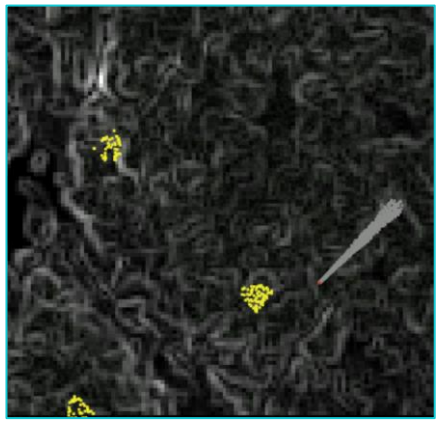
LINKÖPING UNIVERSITY

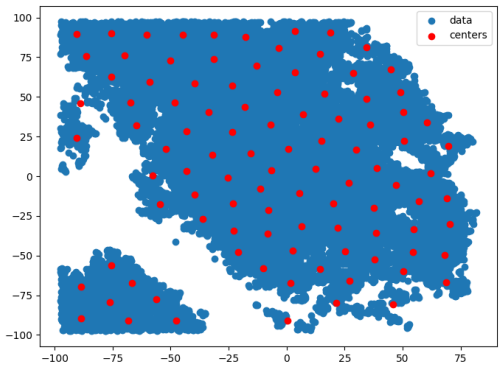
VINNOVA
Sweden's Innovation Agency

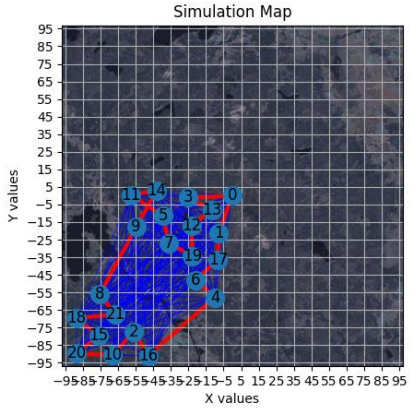
MODPROD

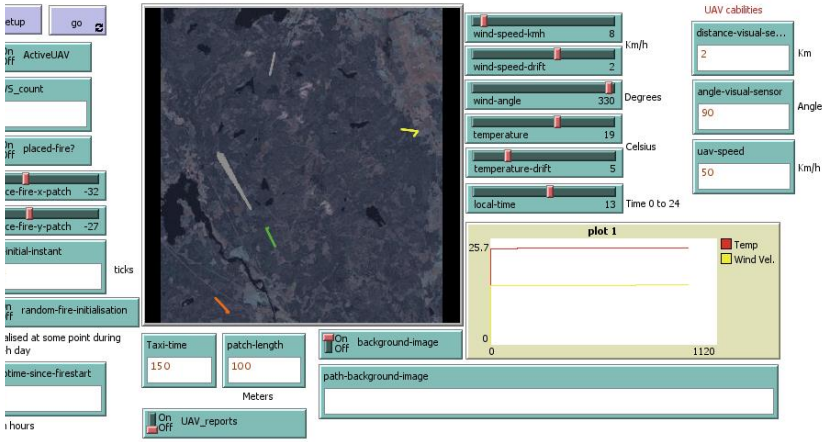
1. Background – The Research Itself

- **System of Systems (SoS)** : Independent systems integrated in such a way that the group value is increased beyond prediction.
- Improve detection by using Unmanned Aerial Vehicles (UAVs):
  - Improve early detection and response.
  - Goal: Requirements for aircraft conceptual design.









1. Background – The Idea

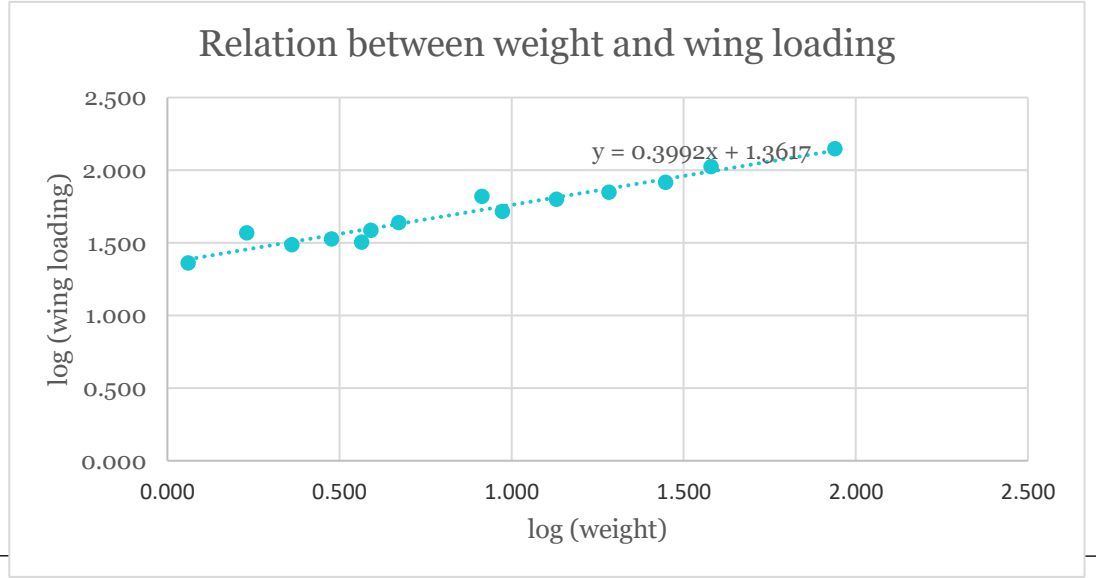
- **From SoS analysis**: Endurance & Payload.
- Systems Engineering Design:
  - Based on our courses.
  - From requirements to High fidelity model.
- Why Machine Learning + Aircraft Design?
  - Size is too small.
  - We don’t use all our data.
  - Faster workflows.
  - Trying to promote it!

2. The Models

- Starting from statistical models:
  - We can't beat birds yet
  - And electric motors
- Basic aircraft relationships:
  - $AR = \frac{b^2}{S} = \frac{Wing\ span^2}{Wing\ Area}$
  - $\frac{L}{D} = \frac{Lift}{Drag} = \frac{Weight}{Thrust}$
  - $\frac{W}{S} = \frac{Total\ Mass}{Wing\ Area}$
- Electric battery:
  - $V = \frac{W}{A} = \frac{Power}{Current}$

Simple script

Name	Weight W[N]	Wing area S[m2]	Wing loading W/S [N/m^2]	Cruise speed v[m/s]
Common tern	1.15	0.05	23	7.8
Dove prion	1.7	0.046	37	9.9
Black-headed gull	2.3	0.075	31	9
Black skimmer	3	0.089	34	9.4
Common gull	3.67	0.115	32	9.2
Kittiwake	3.9	0.101	39	10.1
Royal tern	4.7	0.108	44	10.7
Fulmar	8.2	0.124	66	13.2
Herring gull	9.4	0.181	52	11.7
Great skua	13.5	0.214	63	12.9
Great black-backed gull	19.2	0.272	71	13.6
Sooty albatross	28	0.34	82	14.7
Black-browned albatross	38	0.36	106	16.7
Wandering albatross	87	0.62	140	19.2



2. Propeller Models: Where the Fun Begins

- The diameter of the propeller is needed:
  a. Propeller models are too demanding at this point (Goldstein & Theodorsen).
  b. Propeller diameter quick approximations.
  c. Experimental data for small propellers.
     i. Based on actual aircraft projects.
     ii. Propellers: 18–21-inch diameter with 8–10-inch pitch

$$0=Thrust/(density*RPS^2*(0.101032 -(0.119813*Velocity)/(RPS*Diam)))+Diam^4$$

2. Propeller Models: Where the Fun Begins

- No solution for quintic polynomials:
  - Avoid software specific solvers.
  - Explore numerical methods:
  - Newton-Raphson is an obvious choice, but…
  - Julia Symbolics is perfect for this, especially with E. Halley!

[1] Évariste Galois

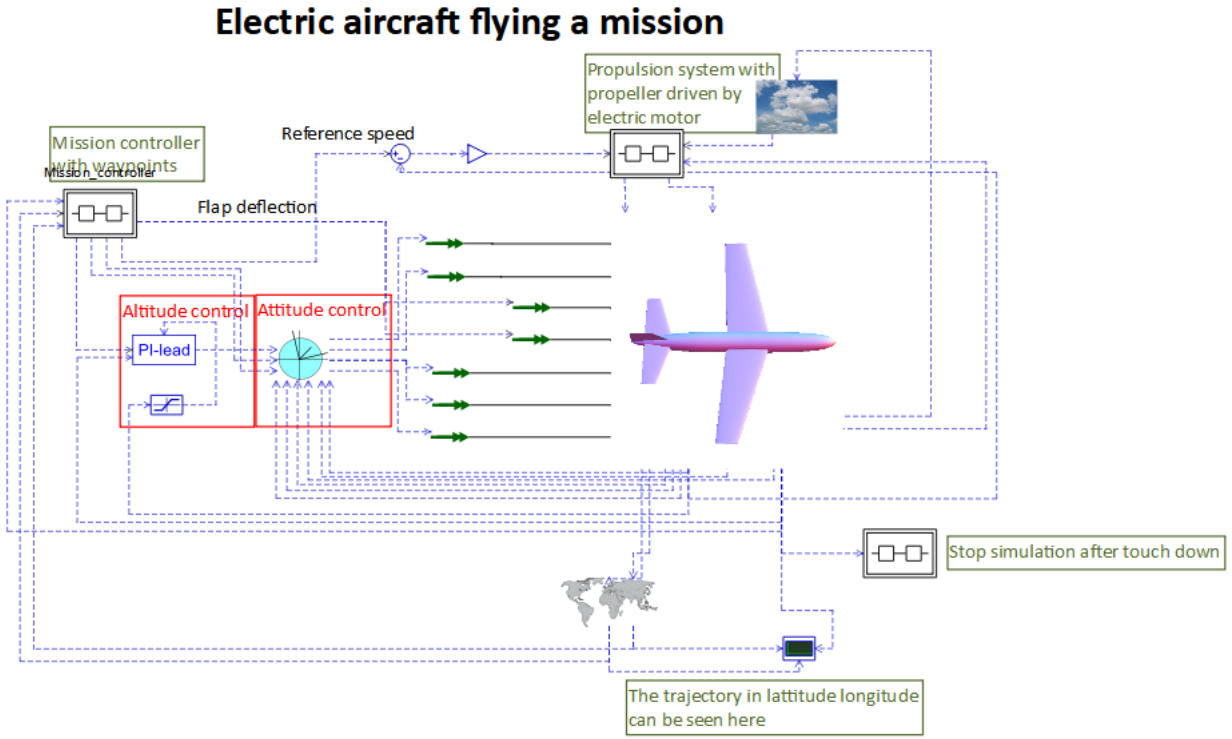
$$x_{n+1} = x_n - \frac{2 \cdot f(x_n) \cdot f'(x_n)}{2\left(f'(x_n)\right)^2 - f(x_n) \cdot f''(x_n)}$$

```
#Halley's Method
using Symbolics
@variables x
f=x^4-Thrust/(((-0.119813)*(Vel/(RPS*x)) + 0.101032)*density*RPS^2);
D_x=Differential(x);
df=expand_derivatives(D_x(f));
dff=expand_derivatives(D_x(df));
```

[1]"Évariste Galois." Wikimedia Commons. Public domain. Accessed 5th Feb 2024. https://commons.wikimedia.org/wiki/File:Evariste_galois.jpg.

3. The Simulation Part

- In-house developed high-performance software for acausal modelling: **Hopsan**.
- High-fidelity aircraft model.
- Model runs with the predictions.
- Several iterations are needed.
- Can be tedious done “by hand”…



4. The Julia Part

- It is easy to use the generated data:
  - Load into data frame
  - Select input/output
  - Select model
  - Fit the model
  - Save the model

```
@time begin
using DataFrames, CSV, DecisionTree, ScikitLearn.CrossValidation, JLD2
train_df     = CSV.read("MODPROD2024/FlightsDataCSV.csv", DataFrame)
y            = train_df[:,"Capacity_Ah"]
X            = Matrix(train_df[:,Not(["Capacity_Ah"])])
model        = RandomForestClassifier(n_trees=1000)
fit!(model,X,y)
save_object("MODPROD2024/AC_Capacity_Predictor.jld2", model)

model        = load_object("MODPROD2024/AC_Capacity_Predictor.jld2")
test_df      = CSV.read("MODPROD2024/test.csv",DataFrame)
X_test       = Matrix(test_df[:,Not(["Capacity_Ah"])])
capacityAh   = predict(model, X_test)
end
```

```
 16.586933 seconds (14.71 M allocations: 980.570 MiB, 4.13% gc time,
 71.46% compilation time: 31% of which was recompilation)
1-element Vector{Float64}:
 81.13960909995001
```

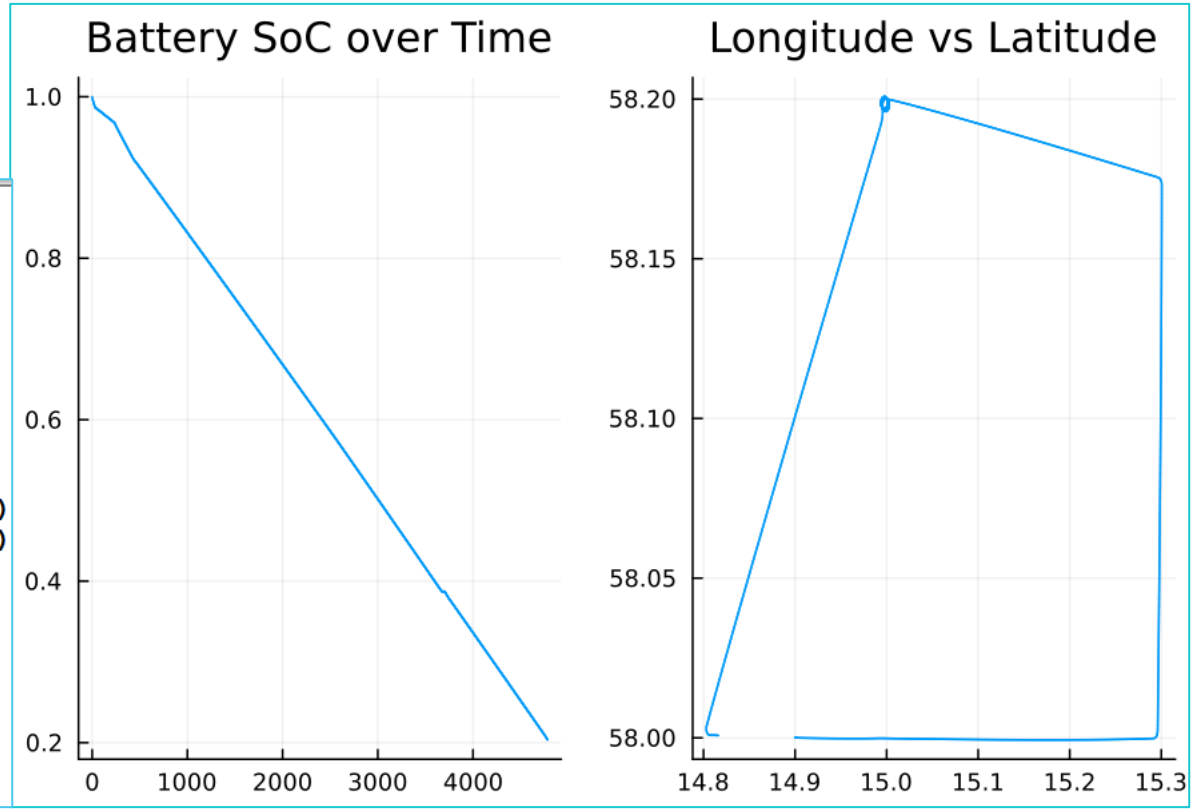
```
  0.328240 seconds (1.31 M allocations: 67.009 MiB, 3.21% gc time)
1-element Vector{Float64}:
 81.26218000266
```

Loiter_s	Payload_kg	Velocity_m_s	Mission_Time_h	Thrust_N	Power_W	Capacity_Ah	Battery_Mass_kg	Motor_Mass_kg	Total_Mass_kg	Structure_Mass_kg	AspectRatio	WingArea_m2	WingSpan_m	WingChord_m	Re	diam
300	8	24.64022	1.323402	11.63166	358.2584	63.49818	3.555898	0.308027	14.82991	2.965981	7	1.406016	3.137214	0.448173	734948.2	0.347935
600	8	24.6649	1.405494	11.69039	360.428	67.84546	3.799346	0.310071	15.13677	3.027354	7	1.410277	3.141964	0.448852	736798.3	0.348333
1200	8	24.715	1.5697	11.81	364.85	76.7	4.2952	0.31425	15.762	3.1524	7	1.4189	3.1516	0.45023	740550	0.34914
300	10	25.76245	1.269383	14.55252	468.6359	65	3.64	0.41343	18.59377	3.718755	7	1.608592	3.355614	0.479373	821915.5	0.366579
600	10	25.78947	1.351474	14.62946	471.6075	70	3.92	0.416303	18.99568	3.799136	7	1.613696	3.360933	0.480133	824081.8	0.367026
1200	10	25.8442	1.515632	14.78631	477.6754	81	4.536	0.422176	19.81503	3.963006	7	1.624068	3.371717	0.481674	828480.5	0.367932
300	12	26.71745	1.226989	17.47651	583.6595	67	3.752	0.525845	22.37115	4.474231	7	1.79564	3.545346	0.506478	900578.7	0.382661
600	12	26.74654	1.309078	17.57244	587.5026	73	4.088	0.52964	22.87224	4.574448	7	1.801556	3.551181	0.507312	903043.2	0.383148
1200	12	26.80551	1.473232	17.76817	595.3561	86	4.816	0.537403	23.89454	4.778909	7	1.813585	3.563018	0.509003	908050.5	0.384135
300	14	27.55264	1.192322	20.40323	702.7033	70	3.92	0.644455	26.1604	5.23208	7	1.970696	3.714145	0.530592	972948.8	0.396857
600	14	27.58361	1.27441	20.51886	707.4802	75	4.2	0.649257	26.76426	5.352851	7	1.977398	3.720455	0.531494	975697.5	0.397379
1200	14	27.64641	1.43856	20.75492	717.2488	89	4.984	0.659087	27.99704	5.599408	7	1.991035	3.733262	0.533323	981285.1	0.398437
300	16	28.29732	1.163137	23.33241	825.3059	76	4.256	0.768647	29.96028	5.992056	7	2.136105	3.866877	0.552411	1040336	0.409618
600	16	28.33002	1.245224	23.46835	831.0736	78	4.368	0.774535	30.6701	6.13402	7	2.143574	3.873631	0.553376	1043358	0.410171
1200	16	28.39635	1.409371	23.74604	842.876	92	5.152	0.786596	32.12005	6.424009	7	2.158777	3.887343	0.555335	1049502	0.411293
300	18	28.97099	1.138028	26.26383	951.1115	76	4.256	0.897937	33.76984	6.753969	7	2.293522	4.006826	0.572404	1103651	0.421074
600	18	29.0053	1.220114	26.42063	957.9229	83	4.648	0.904986	34.5885	6.9177	7	2.301739	4.013997	0.573428	1106936	0.421661
1200	18	29.0749	1.384259	26.74111	971.869	97	5.432	0.919434	36.26165	7.25233	7	2.318472	4.028561	0.575509	1113618	0.422852
300	20	29.58729	1.116059	29.19731	1079.837	80	4.48	1.031938	37.58831	7.517662	7	2.444165	4.136321	0.590903	1163557	0.431757
600	20	29.6231	1.198144	29.37549	1087.741	87	4.872	1.040219	38.51844	7.703687	7	2.453115	4.143888	0.591984	1167096	0.43237
1200	20	29.69578	1.362286	29.73982	1103.934	101	5.656	1.057199	40.42032	8.084063	7	2.47135	4.15926	0.59418	1174299	0.433617

5. Yet Another Simulation Part

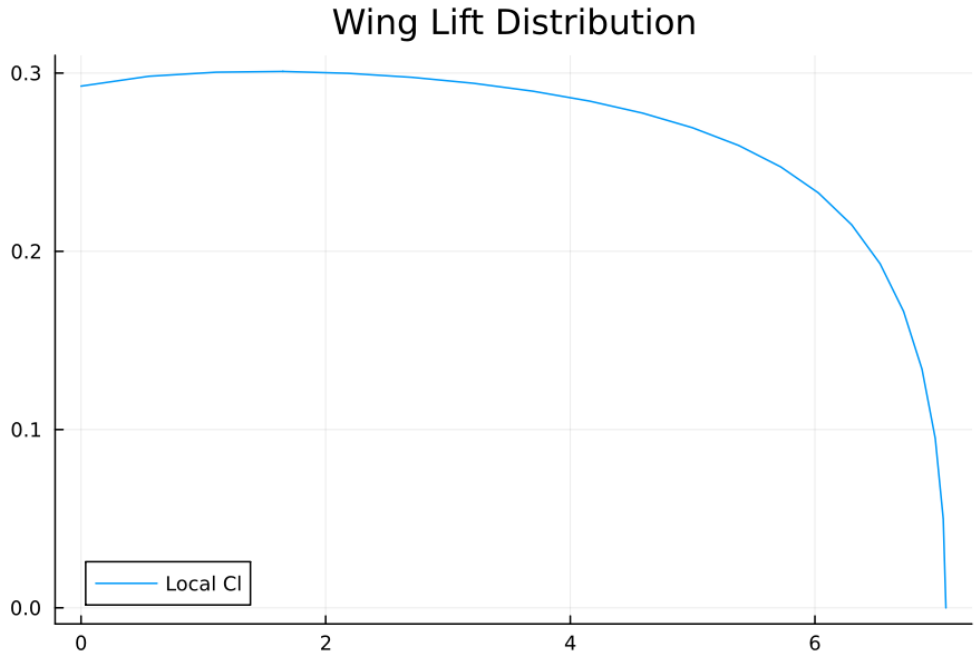
- **Hopsan** is C++ based...
- So, we can call it from Julia as well!!

```
ENV["HOPSANHOME"] = "MODPROD2024/Hopsan"
include("MODPROD2024/hopsan.jl")
using .hopsan
hopsan.loadmodel("MODPROD2024/MODPROD2024.hmf")
hopsan.simulate()

# Retrieve time vector
time_vector = hopsan.gettimevector()

# Retrieve data vector for a specific variable
variable_data = hopsan.getdatavector("Electric_propeller_drive.SoC_Port.Value")
variable_dataB = hopsan.getdatavector("SignalEarthCoordinates.longitude.Value")
variable_dataC = hopsan.getdatavector("SignalEarthCoordinates.latitude.Value")

# Plotting the data

using Plots
p1=plot(time_vector, variable_data, title="Battery SoC over Time");
p2=plot(variable_dataB, variable_dataC, title="Longitude vs Latitude");
plot(p1, p2, layout=(1,2), legend=false)
```



6. Conclusion & Future Work

- Conclusions:
  - Easy to train a model
  - Use of generated data
  - No need of future predictions
  - Simplifies and accelerates the workflow
- Future Work:
  - Extend the data set
  - Include more aerodynamics (Lifting-Line based)
  - Improve Hopsan-Julia connection
  - Hopsan + ModelingToolkit



Questions? jorge.lovaco@liu.se

Thank you for listening!

LINKÖPING UNIVERSITY

VINNOVA
Sweden's Innovation Agency

MODPROD