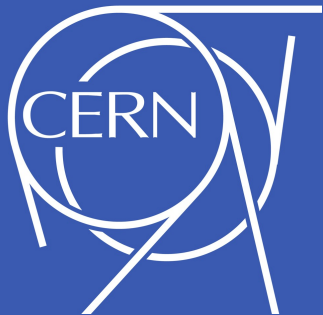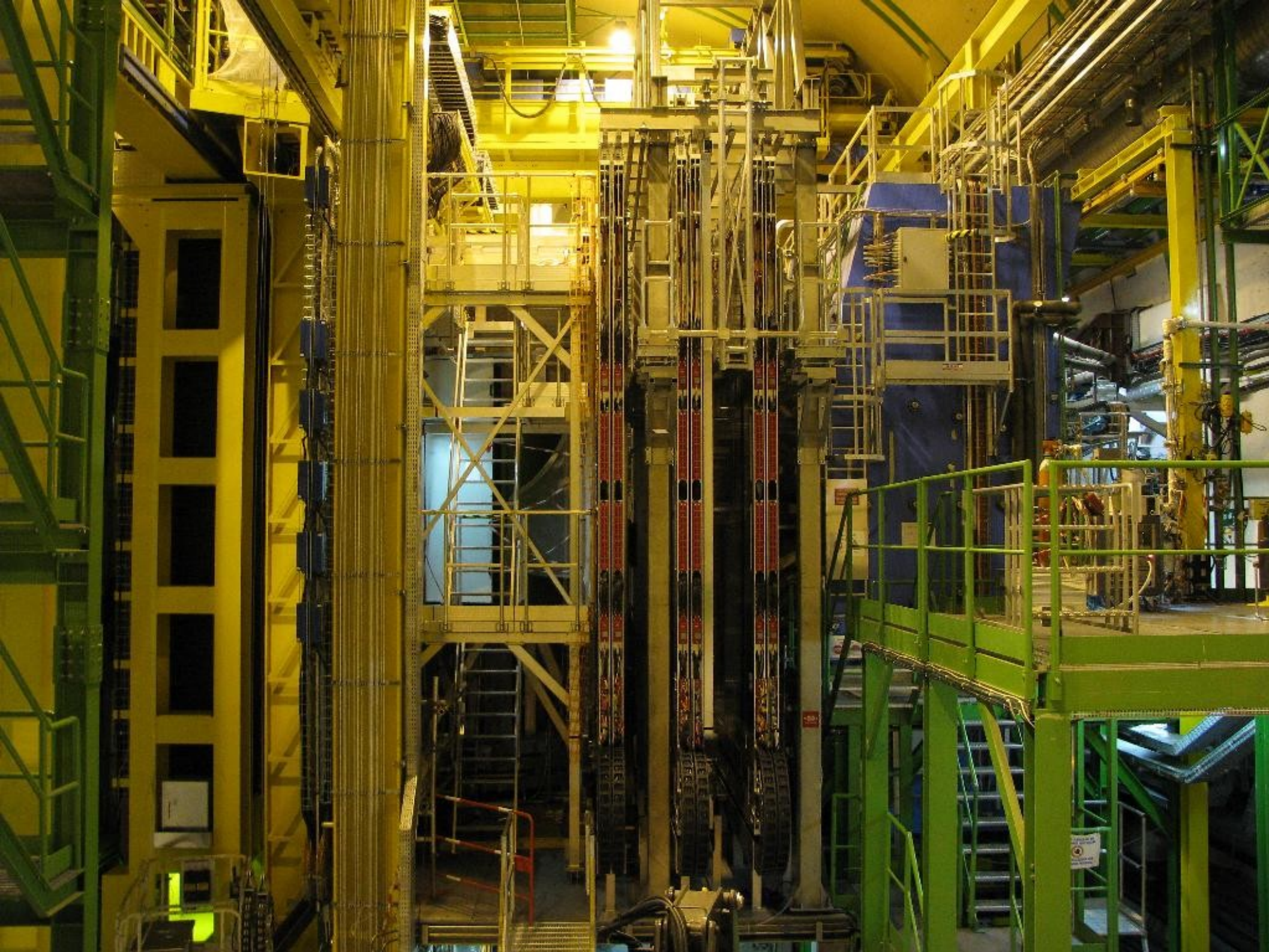# Bringing the power of dynamic languages to hardware control systems

J. M. Caicedo, R. Stoica, N. Neufeld

CERN, European Organization for Nuclear Research

Problems

- Integrating new devices into a DCS can be complicated

    - Learn how the device works

    - Different protocols for communicating with the devices

    - Requires prototyping and experimentation

Problems

- Software developed and maintained by people from multiple backgrounds

- Traditionally using *system programming languages*

  - Complexity

- Not written in isolation: frameworks, build environments ...

Problems

- The development cycle depends on the ease of doing changes.

Edit - build - run   Scripts   Spreadsheets

← Slower   Faster →

If only it could be this easy ...

Planet.numbers

Arial | Negrita | 10 | | | | | | | Ajustar | 1.0 | $ | % | ✓ | ▾ | ◂.00 | .0 ▸ | | Fino | | | | …

Hojas | 20

Planetary Motion

Michael Fowler, UVa

Leapfrog Integration

	G=	20	To get an undistorted picture of the orbit,
delta t=	0.28	turn off automatic scaling, click on the chart
x init=	0	area and stretch it appropriately so that
y_init=	15	one unit in the x-direction on your chart
v x init=	1	looks the same length as one unit in the
v y init=	0	y-direction. To print, in Print Preview turn off
		Scale to Fit Page
		halfway variables

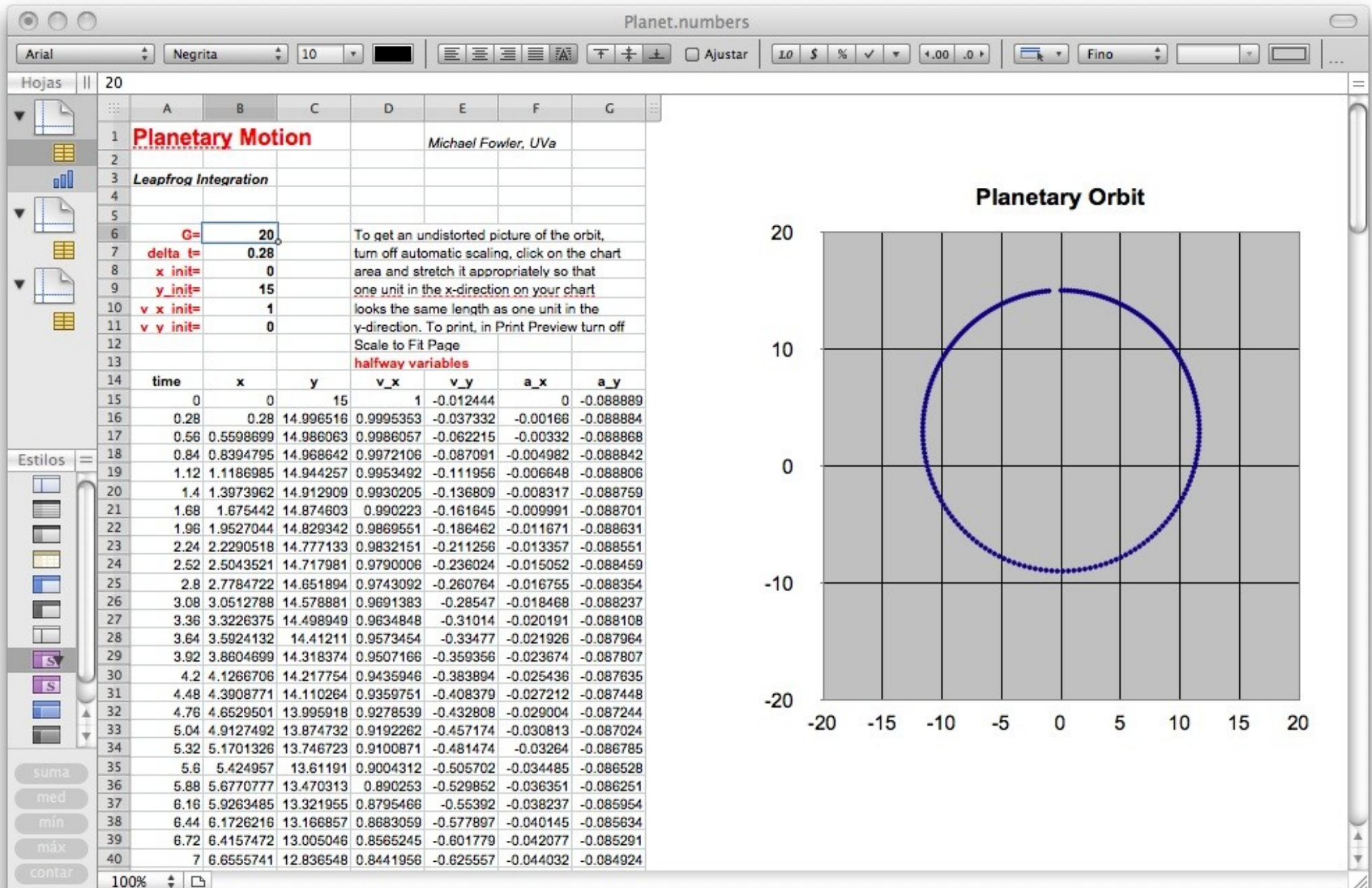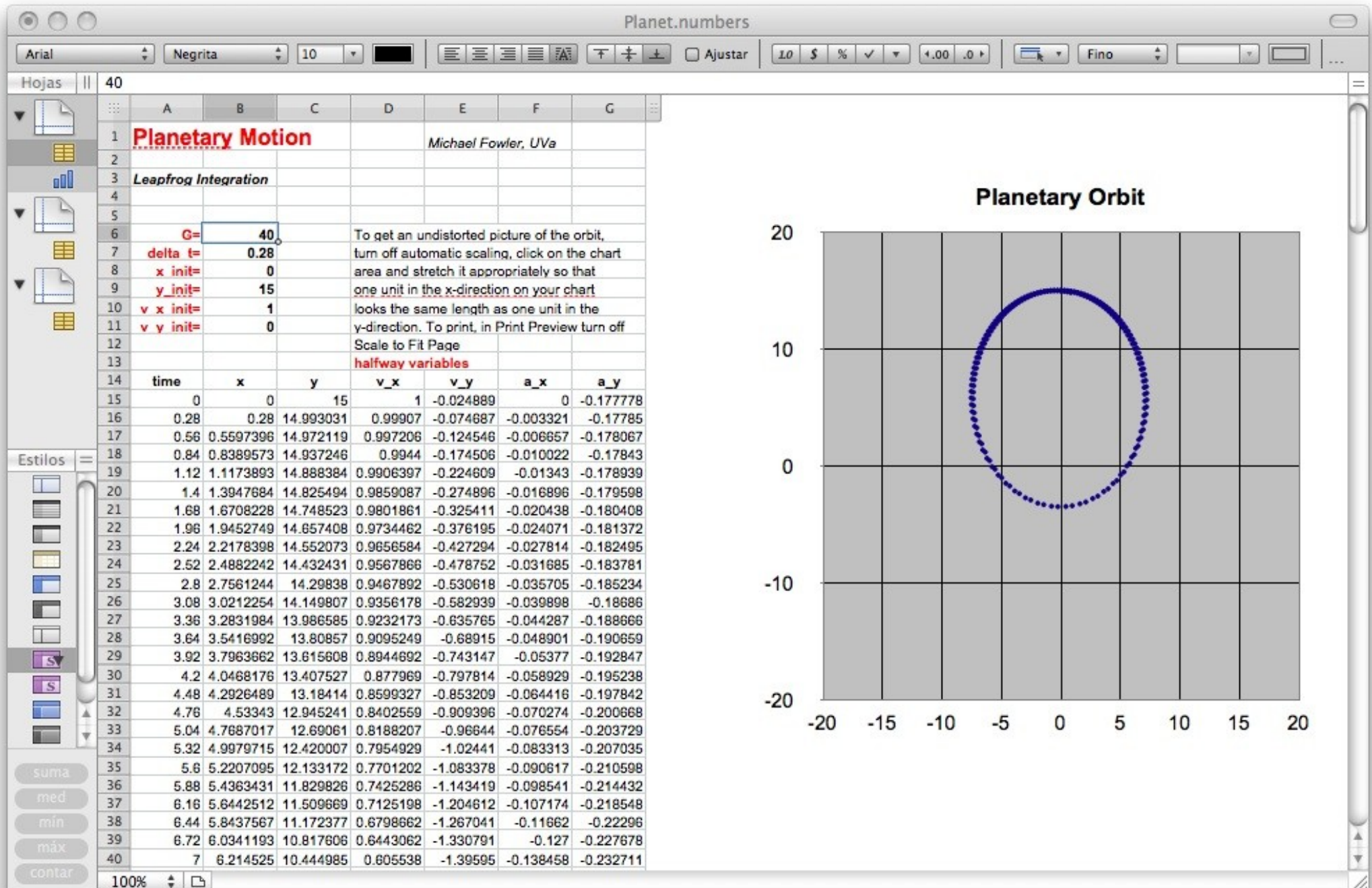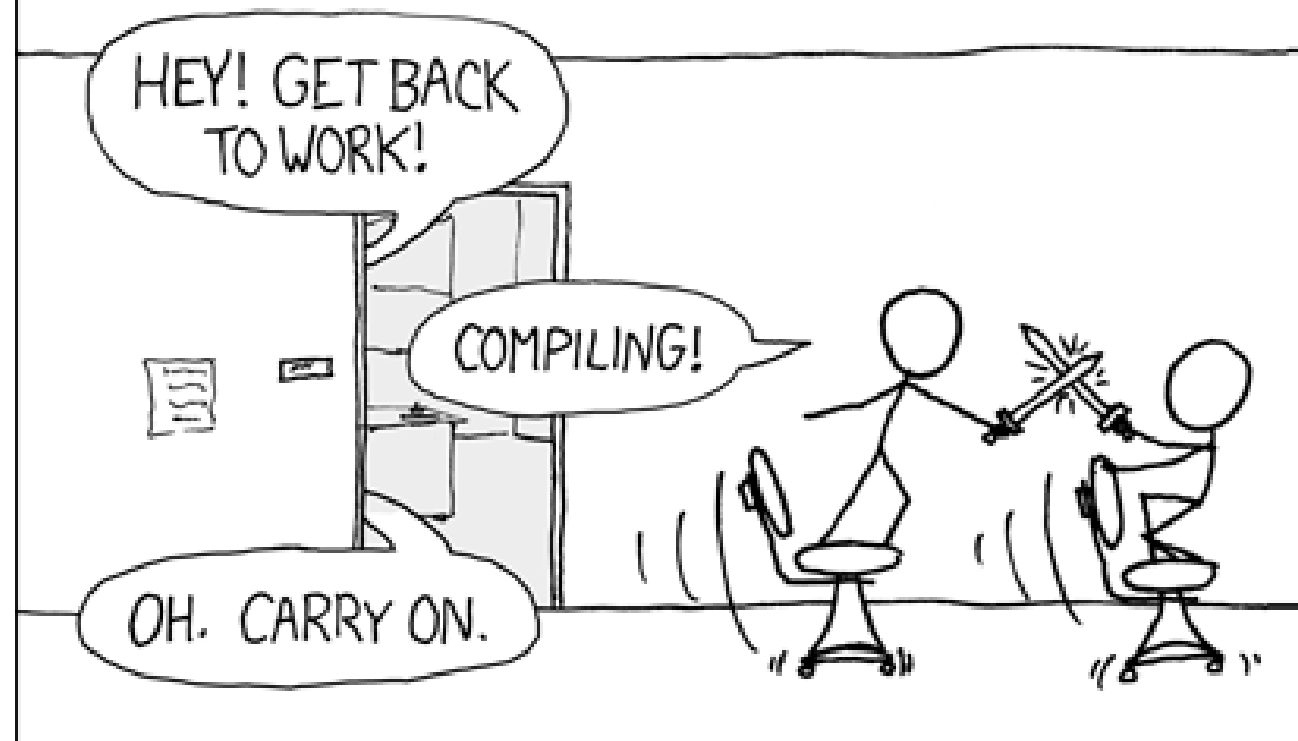| time	x	y	v_x	v_y	a_x	a_y
0	0	15	1	-0.012444	0	-0.088889
0.28	0.28	14.996516	0.9995353	-0.037332	-0.00166	-0.088884
0.56	0.5598699	14.986063	0.9986057	-0.062215	-0.00332	-0.088868
0.84	0.8394795	14.968642	0.9972106	-0.087091	-0.004982	-0.088842
1.12	1.1186985	14.944257	0.9953492	-0.111956	-0.006648	-0.088806
1.4	1.3973962	14.912909	0.9930205	-0.136809	-0.008317	-0.088759
1.68	1.675442	14.874603	0.990223	-0.161645	-0.009991	-0.088701
1.96	1.9527044	14.829342	0.9869551	-0.186462	-0.011671	-0.088631
2.24	2.2290518	14.777133	0.9832151	-0.211256	-0.013357	-0.088551
2.52	2.5043521	14.717981	0.9790006	-0.236024	-0.015052	-0.088459
2.8	2.7784722	14.651894	0.9743092	-0.260764	-0.016755	-0.088354
3.08	3.0512788	14.578881	0.9691383	-0.28547	-0.018468	-0.088237
3.36	3.3226375	14.498949	0.9634848	-0.31014	-0.020191	-0.088108
3.64	3.5924132	14.41211	0.9573454	-0.33477	-0.021926	-0.087964
3.92	3.8604699	14.318374	0.9507166	-0.359356	-0.023674	-0.087807
4.2	4.1266706	14.217754	0.9435946	-0.383894	-0.025436	-0.087635
4.48	4.3908771	14.110264	0.9359751	-0.408379	-0.027212	-0.087448
4.76	4.6529501	13.995918	0.9278539	-0.432808	-0.029004	-0.087244
5.04	4.9127492	13.874732	0.9192262	-0.457174	-0.030813	-0.087024
5.32	5.1701326	13.746723	0.9100871	-0.481474	-0.03264	-0.086785
5.6	5.424957	13.61191	0.9004312	-0.505702	-0.034485	-0.086528
5.88	5.6770777	13.470313	0.890253	-0.529852	-0.036351	-0.086251
6.16	5.9263485	13.321955	0.8795466	-0.55392	-0.038237	-0.085954
6.44	6.1726216	13.166857	0.8683059	-0.577897	-0.040145	-0.085634
6.72	6.4157472	13.005046	0.8565245	-0.601779	-0.042077	-0.085291
7	6.6555741	12.836548	0.8441956	-0.625557	-0.044032	-0.084924

Planetary Orbit

suma | med | min | máx | contar

100%

If only it could be this easy …

Planet.numbers

Arial | Negrita | 10 | ☐ Ajustar | Fino

Hojas | 40

Planetary Motion
Michael Fowler, UVa

Leapfrog Integration

G=	40	To get an undistorted picture of the orbit,
delta t=	0.28	turn off automatic scaling, click on the chart
x init=	0	area and stretch it appropriately so that
y_init=	15	one unit in the x-direction on your chart
v x init=	1	looks the same length as one unit in the
v y init=	0	y-direction. To print, in Print Preview turn off
		Scale to Fit Page
		halfway variables

time	x	y	v_x	v_y	a_x	a_y
0	0	15	1	-0.024889	0	-0.177778
0.28	0.28	14.993031	0.99907	-0.074687	-0.003321	-0.17785
0.56	0.5597396	14.972119	0.997206	-0.124546	-0.006657	-0.178067
0.84	0.8389573	14.937246	0.9944	-0.174506	-0.010022	-0.17843
1.12	1.1173893	14.888384	0.9906397	-0.224609	-0.01343	-0.178939
1.4	1.3947684	14.825494	0.9859087	-0.274896	-0.016896	-0.179598
1.68	1.6708228	14.748523	0.9801861	-0.325411	-0.020438	-0.180408
1.96	1.9452749	14.657408	0.9734462	-0.376195	-0.024071	-0.181372
2.24	2.2178398	14.552073	0.9656584	-0.427294	-0.027814	-0.182495
2.52	2.4882242	14.432431	0.9567866	-0.478752	-0.031685	-0.183781
2.8	2.7561244	14.29838	0.9467892	-0.530618	-0.035705	-0.185234
3.08	3.0212254	14.149807	0.9356178	-0.582939	-0.039898	-0.18686
3.36	3.2831984	13.986585	0.9232173	-0.635765	-0.044287	-0.188666
3.64	3.5416992	13.80857	0.9095249	-0.68915	-0.048901	-0.190659
3.92	3.7963662	13.615608	0.8944692	-0.743147	-0.05377	-0.192847
4.2	4.0468176	13.407527	0.877969	-0.797814	-0.058929	-0.195238
4.48	4.2926489	13.18414	0.8599327	-0.853209	-0.064416	-0.197842
4.76	4.53343	12.945241	0.8402559	-0.909396	-0.070274	-0.200668
5.04	4.7687017	12.69061	0.8188207	-0.96644	-0.076554	-0.203729
5.32	4.9979715	12.420007	0.7954929	-1.02441	-0.083313	-0.207035
5.6	5.2207095	12.133172	0.7701202	-1.083378	-0.090617	-0.210598
5.88	5.4363431	11.829826	0.7425286	-1.143419	-0.098541	-0.214432
6.16	5.6442512	11.509669	0.7125198	-1.204612	-0.107174	-0.218548
6.44	5.8437567	11.172377	0.6798662	-1.267041	-0.11662	-0.22296
6.72	6.0341193	10.817606	0.6443062	-1.330791	-0.127	-0.227678
7	6.214525	10.444985	0.605538	-1.39595	-0.138458	-0.232711

Planetary Orbit

Estilos

suma | med | min | máx | contar

100%

Our approach

- To use dynamic languages for controlling devices

- Integration with the LHCb ECS

Dynamic languages

- Interpreted

- Reflection

  – Modify the program at run time

- Higher level instructions and data types

  – Less verbose

  – Example: string manipulation

Interactive programming

- Receive immediate feedback

- Fast *edit – test – debug* cycle

- Encourages prototyping

Productivity

- **Warning!** Qualitative and anecdotal evidence

- Simplicity

- Useful data structures

- "Do more with less code"

Our approach

- To use dynamic languages for controlling devices

- Integration with the LHCb ECS

SCADA System
PVSS II

DIM - Communication framework

Hardware devices

SCADA System
PVSS II

DIM - Communication framework

Services

DIM
clients

DIM
servers

Hardware devices

PyDIM = Python + DIM

- Wrapper around DIM C library
- Write DIM clients and servers in Python
- Transparent conversion of data types
- Compatible with other DIM clients and servers
- All the DIM features are available

Performance: PyDIM vs C/C++

Is it slow? Yes.

	PyDIM	C/C++
Latency	1980 µs	996 µs
Memory usage	2416 kB	472 kB
Throughput	249 k serv/s	563 k serv/s

Maximum memory used for a DIM server with 25 services

Average throughput of a client receiving messages of different sizes (4k - 512k)

€ 10,00

SONY
MC-30
SONY MICROCASSETTE
A

DURACELL
SIZE D MN1300 LR20
ALKALINE 1,5 Volts

VARTA
HIGH ENERGY

VARTA
HIGH ENERGY

DURACELL

DURACELL
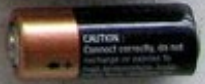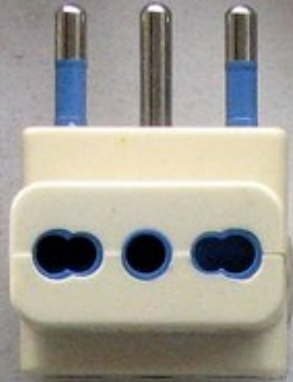
Powersocket

- Control power management devices
  - Telnet: automated interactive session
  - SNMP
- Integrated with the DCS using DIM
- Prototype different alternatives

Conclusions

- Dynamic languages can provide a shorter development cycle

- PyDIM and DIM: enable efficient and reliable communication

- Dynamic languages features have a performance cost

    – Modern implementation techniques