Simplot
Manual de diseño y uso
Estado actual, ejemplos reproducibles y evolución

Proyecto Simusol

9 de octubre de 2026

Alcance y forma de lectura

Este manual describe el código actual de Simplot, no únicamente las prestaciones de la versión histórica. Reúne su filosofía, instalación, formas de entrada, unidades, motores gráficos, interfaz Perl, uso de Gnuplot, limitaciones y posibilidades de ampliación. Los ejemplos de `ejemplos/` son entradas reales ejecutadas por `verificar.pl`; las figuras se generan con ese mismo programa, no son ilustraciones simuladas.

Cada ejemplo gráfico completo se acompaña de su salida real de Gnuplot. Los fragmentos que explican una orden aislada se leen dentro del ejemplo correspondiente; no son documentos ejecutables independientes. En los ejemplos por lotes se reutilizan los mismos dibujos, no una salida distinta del lenguaje.

La referencia de distribución es `2017.0.2.R1017-3`. El manual y sus ejemplos son posteriores a los paquetes publicados de esa revisión: no se supone que ya estén dentro de los binarios descargados. Las capacidades descritas se contrastan con los archivos del árbol actual. Las cifras de pruebas de documentos anteriores se conservan como registros históricos, no como promesa de cobertura exhaustiva.

Si necesita...	Empiece por...
Decidir qué herramienta usar	Capítulos de filosofía y comparación; distinción entre cálculo y dibujo.
Dibujar ahora	Ejemplos de tablas, RESULTADOS, PARAPLOT y un solo punto.
Entender y escribir PARAPLOT	Capítulo 3: conceptos, órdenes, composición, compatibilidad y evolución.
Integrar Simplot en un programa	Contrato DAT y API por contexto.
Mantener o extender el sistema	Arquitectura, límites y plan de evolución.
Reproducir el manual	`make-Cdoc/manual`; requiere además LaTeX y latexmk.

Índice general

1. Filosofía y arquitectura **1**

1.1. Una descripción de gráficos separada del cálculo 1

1.2. Principios que conviene preservar 2

1.3. Unificar el motor conservando las prestaciones 3

1.4. Recorrido de una ejecución 3

2. Instalación, documentación y aspectos técnicos **5**

2.1. Instalar 5

2.1.1. Paquetes binarios 5

2.1.2. Desde fuentes (recomendado para obtener el código actual) 5

2.2. Construir y ver la documentación 6

2.3. Técnicos 6

2.3.1. Construir manualmente desde un árbol de fuentes 6

2.3.2. Construir un paquete binario desde fuentes 7

2.3.3. Instalar desde un paquete construido 8

2.3.4. Dependencias 8

3. PARAPLOT: lógica del lenguaje y forma de usarlo **10**

3.1. Qué es PARAPLOT 10

3.1.1. Dos misiones y nombres de las secciones 10

3.2. Empezar por RESULTADOS: PARAPLOT ultrasimple 11

3.3. Los objetos: fuente, columna, serie, gráfico y panel 12

3.4. Cómo se lee una orden 13

3.5. Un pedido completo, explicado por decisiones 14

3.6. Superponer no es poner en paneles 14

3.7. Varias fuentes: nombrar la propia implícita y la externa 15

3.8. PARAPLOT independiente: declarar dónde están los datos 16

3.9. Ejes, rótulos, unidades y encuadre 17

3.10. Un punto, una trayectoria y los límites de error 18

3.11. El mismo pedido en .ds, DAT y el pipe . . . 18
3.12. Compatibilidad: qué existe hoy y qué debe conservarse . . . 19
3.13. Cómo será más claro, potente y expresivo . . . 20

4. Uso concreto: de una tabla a una presentación 22
4.1. Preparar un lugar de trabajo . . . 22
4.2. Sin instrucciones: una tabla numérica . . . 22
4.3. Elegir columnas de forma interactiva . . . 23
4.4. PARAPLOT: vista temporal y espacio de estados . . . 24
4.5. Superponer dos fuentes: cálculo propio y mediciones . . . 25
4.6. PARAPLOT: barras de error con límites recibidos . . . 27
4.7. RESULTADOS: la sintaxis ultrasimple de PARAPLOT . . . 29
4.8. Unidades: origen, presentación y temperatura . . . 30
4.9. Un solo punto: graficar el espacio de coordenadas . . . 31
4.10. CSV, fechas, horas y zona horaria . . . 31
4.11. Comparar corridas, usar logaritmos y exportar cuatro formatos . . . 32
4.12. Conservar descripciones históricas . . . 34

5. Referencia del modelo y de la interfaz Perl 36
5.1. El DAT como frontera entre programas . . . 36
5.2. Qué se expresa en .ds y qué requiere DAT . . . 37
5.3. API por contexto: uso en lotes y servicios . . . 37
5.4. Alcance de la reentrancia . . . 38

6. Gnuplot: motor gráfico, archivos y personalización 40
6.1. Qué aporta y qué no expone Simplot . . . 40
6.2. Archivos de trabajo . . . 40
6.3. Volver a ejecutar o adaptar un guion . . . 41
6.4. Ventanas, ejecución sin pantalla y seguridad . . . 41
6.5. Diagnóstico por capas . . . 41

7. Limitaciones actuales y extensiones posibles 43
7.1. Matriz de alcance . . . 43
7.2. Procesamiento numérico: ampliarlo en Tabular2 . . . 44
7.3. Extensiones del graficador, sin cambiar su responsabilidad . . . 44
7.4. Trabajo de infraestructura y seguridad . . . 45

8. Comparación con otras herramientas 46

8.1. Elegir por el trabajo, no por una lista de nombres 46
8.2. Nivel de abstracción respecto de PARAPLOT 47
8.2.1. De PARAPLOT al dibujo: una etapa posterior 49
8.2.2. Cómo leer la comparación de las otras herramientas 49
8.2.3. Qué significa la gramática de Observable Plot 50
8.3. D3 en Puyuspa 50
8.4. Programas del propio árbol: no confundir los nombres 51
8.5. Qué adoptar de esas comparaciones 51

9. Mapa del código, corpus y mantenimiento **52**
9.1. Cómo mantener documentación y ejemplos coherentes 52
9.2. Documentación complementaria 53

Capítulo 1

Filosofía y arquitectura

1.1. Una descripción de gráficos separada del cálculo

Simplot convierte tablas existentes y una descripción de lo que se quiere mostrar en gráficos reproducibles. Su valor no es sustituir el lenguaje completo de Gnuplot: es reconocer los datos del proyecto, conservar sus nombres y unidades y permitir que una descripción de resultados acompañe al sistema físico sin atar el dibujo a un simulador determinado.

Esa descripción se expresa en PARAPLOT. RESULTADOS es su sintaxis ultrasimple: una lista de variables, con atributos opcionales, abrevia un pedido que también puede detallarse con fuentes, series, ejes y paneles. No se trata de lenguajes ni procesos de dibujo independientes.

1. Un productor obtiene los valores: Siru, Calcula, Tabular2, Radia, SCEPTRE, un instrumento, una planilla u otro programa.
2. Una tabla guarda las muestras. Sus columnas pueden ser tiempo, estados, parámetros o cualquier magnitud numérica.
3. Una descripción selecciona fuentes, coordenadas, unidades de presentación y organización de gráficos.
4. Simplot valida esa descripción y prepara las tablas y el guion de dibujo.
5. Gnuplot representa el resultado y escribe las imágenes.

No hay obligación de que exista tiempo. La primera columna de una tabla automática se usa como abscisa por convención, no porque Simplot la identifique como tiempo. Una fila es un estado; varias filas pueden representar una trayectoria, un barrido o mediciones sin orden temporal. Esta interpretación pertenece al problema y debe hacerse explícita en los rótulos.

Tampoco hay obligación de que un archivo `.simul` contenga muchas filas. Si contiene una sola, puede corresponder al cálculo de un punto del sistema, no a una simulación temporal. Se puede representar ese punto en el espacio de estados, eligiendo sus variables como coordenadas: por ejemplo, ubicar un estado de aire húmedo en un diagrama psicrométrico. Las proyecciones entre pares de variables muestran ese mismo estado desde distintas coordenadas; las curvas de referencia del diagrama, si se quieren incluir, requieren sus propios datos o cálculos, no se deducen de la única fila.

Responsabilidad	Componente actual
Resolver ecuaciones	Calcula, usado por Siru o de forma independiente; no Simplot.
Simular sistemas	Siru, que usa Calcula, o SCEPTRE como alternativa.
Producir, interpolar o transformar tablas	Tabular2; Radia y otras fuentes también producen datos para Simplot.

Responsabilidad	Componente actual
Interpretar .ds/.d y PARAPLOT	slex, mediante `d2datdinc`; contrato compartido con Siru.
Leer tablas y convertir unidades	Tabular2 mediante `interpola--exportar-simul`; libunidades y GNU units para las unidades.
Elegir series, validar ejes y componer paneles	Simplot.
Trazar líneas, puntos y exportar imágenes	Gnuplot.
Construir, transferir e instalar paquetes	SummaPack y los objetivos de distribución del paquete.

Siru es una alternativa a SCEPTRE y utiliza Calcula para resolver las ecuaciones; Calcula también puede utilizarse de forma independiente. Ninguno de ellos es un requisito para Simplot. Cualquier tabla de datos numéricos en un formato admitido puede graficarse, provenga de Calcula, Tabular, Radia, mediciones, una planilla u otra fuente. Tampoco se necesita ejecutar Dia. Que el archivo se llame `.simul` no obliga a usar un simulador determinado. Si se entrega un DAT gráfico completo, ni siquiera es necesario volver a analizar el .ds para ese dibujo.

1.2. Principios que conviene preservar

Separación acordada el 11/10/2026, en implementación. UNIDADES y RESULTADOS se interpretan en Slex (`ds2dat/d2datdinc`), al construir el .dat desde el .ds, antes de simular con Siru: unidades de origen, magnitudes, selección, nombres y unidades de salida. Se convierten los datos de entrada; los pedidos de salida quedan preparados para aplicarlos cuando existan los resultados. Simplot recibe datos y metadatos ya resueltos y usa PARAPLOT para dibujar. Si recibe directamente un .ds, utiliza ese mismo procesamiento previo, sin duplicar las reglas físicas ni exigir una simulación para una tabla externa. Las conversiones siguen usando Unidades y Tabular2. Las descripciones posteriores del tratamiento histórico en Simplot son compatibilidad pendiente de migrar, no la arquitectura definitiva.

Excepción futura, no prioritaria. Sólo un campo `RESULTADOGRAFICO` permitiría a Simplot intervenir en las unidades, después de Siru y exclusivamente para presentar el gráfico. No modificaría el modelo, los datos del cálculo ni la tabla original. Por ahora se documenta esta posibilidad sin implementarla; no es sintaxis ejecutable ni un sinónimo de RESULTADOS.

Las definiciones físicas se trasladan desde los archivos de unidades de Dia a `library.siru`. FORMAS conserva la referencia al modelo, la rama y la geometría, sin repetir unidades. La prioridad es: unidad junto al número, UNIDADES, modelo de FORMAS y, en ausencia de declaraciones, SI. La gramática única y el estado de la transición se documentan en `slex/d2datdinc/doc/GRAMATICA.md`, sección 3.9.

- Separar datos, metadatos y presentación. Cambiar de Celsius a Fahrenheit en un eje no debe modificar el archivo original.
- Hacer explícita la selección de corridas o bloques, en lugar de mezclarlos silenciosamente.
- Rechazar modelos inconsistentes antes de producir un gráfico engañoso: columnas inexistentes, unidades incompatibles y logaritmos de valores no positivos.
- Mantener cada trabajo independiente. No reutilizar variables globales del motor histórico entre clientes.

- Conservar guiones y tablas de salida para inspeccionar cómo se obtuvo una imagen.
- Tabular2 realiza la interpolación y el procesamiento numérico general; Simplot representa los resultados. Reutilizar esas operaciones, con pruebas propias, en lugar de repetirlas dentro del graficador.

1.3. Unificar el motor conservando las prestaciones

El objetivo es un único motor de Simplot, reentrante, que reúna las prestaciones de las dos implementaciones actuales. Las diferencias siguientes describen el trabajo pendiente de unificación, no dos motores que se quiera mantener como diseño definitivo.

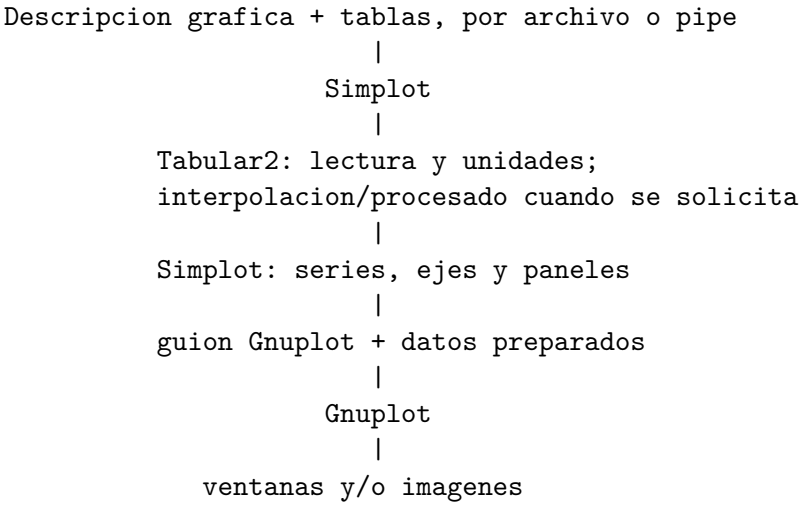
Prestaciones históricas que conservar	Base estructurada que completar
Lenguaje acumulado a lo largo del desarrollo original.	Fuentes, series, ejes, paneles y salida con validación explícita.
RESULTADOS, PARAPLOT histórico, interacción y modo automático.	Subconjunto de PARAPLOT traducido al modelo común, RESULTADOS, DAT tipado y proyecciones por pares.
Conserva selección, referencias funcionales y familias de archivos históricas.	Permite ejes logarítmicos, temporales e invertidos, bloques y formatos mediante DAT.
Internamente conserva estado global; se aísla en un proceso por trabajo.	API por contexto; no modifica el contexto del llamador.
PNG/EPS y archivos históricos de trabajo.	PNG/EPS/PDF/SVG, tablas normalizadas y guion auditable.

La unificación debe traducir las entradas históricas a un modelo común, incorporar sus capacidades todavía ausentes y comprobarlas con el corpus antes de retirar el camino anterior. El usuario no debería elegir entre motores. Hasta completar ese trabajo, los ejemplos incluyen los ajustes transitorios necesarios; las limitaciones se describen por prestación y no como lenguajes separados.

1.4. Recorrido de una ejecución

Este recorrido es la vista de Simplot del mapa global `siru/MAPA-FLUJOS.txt`, que detalla las entradas, salidas y archivos de todos los paquetes. Hay varias entradas alternativas; no es obligatorio recorrer la cadena de simulación para dibujar una tabla.

Origen	Camino hacia Simplot
Simulación con Siru	`.d/.ds` → slex (`d2datdinc`) → modelo para Siru y Calcula. Siru controla la integración y consulta a Calcula por pipe; al terminar envía descripción gráfica y tabla a `simplot--stdin`, salvo `-nosimplot`.
Cálculo independiente	Calcula resuelve puntos o series sin Siru; sus resultados tabulados pueden entregarse a Simplot.
Simulación con SCEPTRE	Camino alternativo a Siru; sus resultados pueden convertirse en tablas admitidas por Simplot. No es una dependencia del graficador.
Otras fuentes	Tabular, Radia, instrumentos, planillas u otros programas suministran tablas. No necesitan Siru, Calcula ni SCEPTRE.
Descripción gráfica por archivo	`.d/.ds`/PARAPLOT → lector slex → descripción JSON. También puede recibirse directamente un `.dat` gráfico con referencias a sus tablas.



La descripción y los datos son entradas distintas aunque viajen juntos por el pipe de Siru. En el modo automático basta una tabla: Simplot selecciona los gráficos sin exigir PARAPLOT. Dibuja, por su parte, representa el esquema del circuito; no interviene en el trazado de sus resultados numéricos.

Estado actual. El pipe Siru–Simplot transporta una cabecera JSON con la descripción gráfica y la tabla completa. Siru todavía guarda resultados y Simplot prepara copias internas; las tablas externas de comparación siguen leyéndose por ruta. Gnuplot abre ventanas o escribe imágenes según las opciones actuales. Por tanto, este enlace no significa que toda la cadena funcione ya sin archivos.

Interfaz acordada, pendiente de completar. El mapa global distingue el objetivo común: entrada completa por archivo o stdin, resultados numéricos por stdout y diagnósticos por stderr; gráficos en X por defecto, `-noX` para no mostrarlos y `-save` para guardar SVG. No debe confundirse esta política con las opciones ya implementadas.

Capítulo 2

Instalación, documentación y aspectos técnicos

2.1. Instalar

2.1.1. Paquetes binarios

Para instalar la versión binaria publicada de Simplot en Debian/Ubuntu:

```
curl -fsSLO https://www.simusol.org/downloads/installsumma.sh
sh installsumma.sh --para deb simplot
```

El gestor instala las dependencias declaradas y permite quitar Simplot con `aptremovesimplot`. También se publican formatos RPM y TGZ, que deben elegirse según la distribución; para quitar un TGZ se usa `instalar-tgz`. La versión binaria estable puede ser anterior al código fuente actual.

2.1.2. Desde fuentes (recomendado para obtener el código actual)

El recorrido previsto descarga las fuentes de Simplot y de las dependencias propias, construye paquetes en la máquina destinataria y los instala con el gestor local; las dependencias externas se obtienen mediante ese gestor:

```
curl -fsSLO https://www.simusol.org/downloads/installsumma.sh
sh installsumma.sh --para fuentes --inestable simplot
```

Ejecutar las órdenes en un directorio donde se pueda guardar el script, sin sobrescribir una copia propia modificada. Se necesita curl y acceso a Internet; el instalador solicita los permisos necesarios. Puede examinarse el recorrido antes de instalar con `shinstallsumma.sh--parafuentes--inestable--pruebasimplot`. En este modo no se admite `--prefijo`: el paquete local se instala en `/usr` mediante el gestor. Si SummaPack ya está instalado, también se puede usar `bajar-paquetes--instalar--parafuentes--inestablesimplot`.

El nombre de la orden importa: `summapacksimplot` no está implementado. Actualmente `summapack` informa la versión y ubicación de las herramientas; la instalación la realizan `installsumma.sh` y `bajar-paquetes`.

Estado comprobado el 12 de octubre de 2026: el modo `--prueba` resuelve la cadena publicada de SummaPack, Unidades, Slex, ExprTk, Tabular2 y Simplot, y enumera las dependencias

externas. En el árbol SVN de SummaPack, Debian/Ubuntu construyen e instalan paquetes DEB locales; Fedora y openSUSE construyen un DEB intermedio, lo convierten en RPM nativo y lo instalan con su gestor. ExprTk y Unidades se probaron en Debian 13, Ubuntu 24.04, Fedora 44 y openSUSE. La cadena completa hasta Simplot se construyó e instaló desde fuentes en contenedores Fedora 44 y Debian 13; aún faltan Ubuntu 24.04 y openSUSE. Arch todavía no tiene este backend. Tampoco se publicó el bootstrap actualizado: no ejecutar la orden pública `--parafuentes` suponiendo que ya construye paquetes. El instalador no recurre a `makeinstall` directo ni soluciona una dependencia faltante omitiéndola.

2.2. Construir y ver la documentación

Desde la raíz de las fuentes:

```
make -C doc/manual check
make -C doc/manual
xdg-open doc/manual/simplot.pdf
```

El primer objetivo ejecuta los ejemplos en un temporal y verifica 26 condiciones. El segundo genera las figuras en `doc/manual/.build/figuras`, compila LaTeX y deja `doc/manual/simplot.pdf`. La última orden abre el visor del escritorio; no se necesita para construir el PDF ni en un servidor sin entorno gráfico. También puede abrirse ese archivo desde un explorador de archivos.

Las fuentes, ejemplos y reglas están versionados; el PDF es generado. Para compilar se necesitan `latexmk`, `texlive-latex-extra`, `texlive-lang-spanish` y `lmodern`, además de las dependencias de ejecución que generan las figuras. Las pruebas usan el código del árbol y los ejecutables instalados de slex, Tabular y Gnuplot. Las salidas temporales se retiran; las entradas originales no se alteran.

La instalación copia fuentes y ejemplos bajo `PREFIX/share/doc/simplot/doc/manual`, y el PDF si ya fue generado; excluye los auxiliares de `.build`. No se presupone que todos los binarios publicados traigan el PDF. Los comandos de construcción se ejecutan en una copia escribible de las fuentes, no en la documentación instalada del sistema.

2.3. Técnicos

2.3.1. Construir manualmente desde un árbol de fuentes

```
cd ~/svn/siru/simplot
make
make check
make install PREFIX="$HOME/.local"
export PATH="$HOME/.local/bin:$PATH"
```

`make` comprueba dependencias y sintaxis; no hay código nativo que compilar ni etapa `configure`. El Makefile actual es el de la raíz. No ejecutar el antiguo `Makefile.PL`: pertenece a otra organización y puede reemplazar el Makefile que se necesita hoy. Las comprobaciones de presencia no sustituyen la comprobación de versiones de las dependencias.

Para instalar para todos los usuarios, ejecutar `sudomakeinstall`, que usa `/usr/local`. La instalación copia archivos; no descarga dependencias ni ejecuta pruebas como administrador.

Con un prefijo privado, el programa instalado encuentra sus módulos relativos a ese prefijo. Un programa Perl propio debe añadir su `share/perl5` a `@INC` si no está en las rutas habituales.

```
make install PREFIX=/usr DESTDIR=/tmp/simplot-paquete
make uninstall PREFIX=/usr DESTDIR=/tmp/simplot-paquete
make distcheck
```

`PREFIX` es el destino lógico; `DESTDIR` agrega una raíz de preparación. Usar los mismos valores al desinstalar. Se registra un manifiesto de archivos, huellas y directorios creados: `share/simplot/install-manifest.json`. La desinstalación conserva archivos modificados o ajenos y comunica el conflicto, en vez de borrarlos por conjetura. Sin manifiesto no se intenta adivinar una instalación vieja. Para un DEB instalado mediante el gestor de paquetes, desinstalar mediante ese gestor, no con `makeuninstall`.

`distcheck` crea una distribución temporal, ejecuta las pruebas y verifica instalación, dibujo y desinstalación, incluyendo prefijos con espacios y DESTDIR. No instala en el sistema anfitrión. Algunas pruebas de integración se omiten cuando faltan los proyectos vecinos; deben leerse los mensajes SKIP, no confundirlos con pruebas ejecutadas. No se admiten instalaciones/desinstalaciones concurrentes sobre el mismo prefijo.

2.3.2. Construir un paquete binario desde fuentes

```
./hacer-pkg
make subir PRUEBA=1
make bajar PRUEBA=1 PARA=deb CANAL=inestable
```

El constructor usa la descripción `simplot.summa` y SummaPack. Produce fuentes, descripción versionada, DEB, TGZ y RPM cuando está disponible su herramienta de construcción; no instala ni publica por el solo hecho de construir. La lista `../subir.lista` determina los artefactos exactos de publicación. `PRUEBA=1` inspecciona sin transferir.

Orden real	Efecto
`makesubir`	Publica la selección como inestable, no promueve a estable.
`makebajarPARA=fuentes`	Descarga fuentes; no instala. El canal predeterminado es estable.
`makebajarPARA=debCANAL=inestable`	Descarga binarios según el catálogo disponible.
`DESCARGAS=/ruta`	Selecciona destino de descarga.
`SUMMAPACK_BIN=/ruta`	Permite seleccionar las herramientas de transferencia.

La configuración de transferencia reside en `~/.config/simusol/subir`: servidor, usuario publicador, directorio remoto de descargas y URL pública. No guardar contraseñas o claves privadas en el repositorio. Véanse `doc/PAQUETES.md` y `scripts/paquetes.pl` para nombres de variables y validaciones. MD5 permite detectar alteraciones accidentales; no reemplaza una firma de autenticidad.

Al cierre de este manual, los cinco artefactos de Simplot -3 se publicaron en https://www.simusol.org/downloads/simplot/otros/. Todavía faltaba publicar allí las dependencias slex 1.0-25 y Tabular 2.0-16. Tampoco había manifiesto estable del proyecto Simplot: para la primera descarga inestable se documentó la selección explícita siguiente. No es una instalación completa con dependencias resueltas:

```
bajar-paquetes --para deb --inestable --sin-dependencias simplot/simplot
```

Este estado de publicación es una fotografía fechada, no una garantía permanente del contenido del servidor. No reconstruir ni sobrescribir la misma versión publicada para incorporar cambios nuevos; corresponde preparar una versión posterior.

2.3.3. Instalar desde un paquete construido

Con el DEB construido y todas las dependencias del proyecto disponibles en versiones compatibles, ejecutar desde el directorio que lo contiene:

```
sudo apt install ./simplot_2017.0.2.R1017-3_all.deb
```

Sustituir el nombre por el archivo exacto producido, no por un patrón que seleccione varias versiones. APT resuelve las dependencias de sus repositorios, pero no conoce por sí solo los paquetes privados de Simusol: si no están instalados, incluir sus DEB compatibles en la misma orden o instalarlos mediante SummaPack. No usar `dpkg--force-depends`. Una instalación con `dpkg-i` solamente no descarga lo que falta.

En una distribución RPM usar su gestor, por ejemplo `sudodnfinstall./simplot-VERSION.noarch.rpm` o `sudozypperinstall./simplot-VERSION.noarch.rpm`, con el nombre real y dependencias compatibles. Un TGZ del proyecto se instala con `instalar-tgz`, no con una extracción indiscriminada sobre `/`; consultar `instalar-tgz--help`. Para retirar un paquete binario usar el mismo gestor que lo instaló, no `makeuninstall`.

2.3.4. Dependencias

Familia	Dependencias y función
Paquetes del proyecto Simusol	`d2datdinc>=1.0-25` de slex interpreta .ds/PARAPLOT; `tabular>=2.0-16` aporta `interpola` para tablas y unidades. Sus bibliotecas transitivas incluyen libslex, libtabular2, libunidades y mem-util. No confundir el proyecto Tabular2 con el nombre del ejecutable.
Software externo	Perl ejecuta Simplot; Gnuplot 5 o posterior dibuja. GNU units participa a través de libunidades. Para ventanas debe estar disponible un terminal gráfico como X11; `gnuplot-nox` sirve para lotes sin ventanas.
Módulos Perl	JSON::PP pertenece al Perl base; Path::Class y Capture::Tiny son externos y se distribuyen como `libpath-class-perl` y `libcapture-tiny-perl` en Debian/Ubuntu. Los módulos SimPlot y Simusol usados por el programa están dentro del paquete.
Construcción de paquetes	SummaPack, Subversion, dpkg-buildpackage, debhelper, fakeroot, make y el entorno de pruebas. Las dependencias de construcción no son necesariamente requisitos de ejecución.
Documentación	LaTeX, español, lmodern y latexmk, más el entorno de ejecución para producir figuras. No hacen falta para usar una instalación de Simplot.

`simplot.summa` es la descripción de empaquetado: DEPENDS declara ejecución y BUILDDEPENDS construcción; las bibliotecas nativas completan dependencias transitivas en sus propios paquetes. Los nombres RPM/TGZ pueden diferir y SummaPack mantiene sus equivalencias. No alcanza con copiar un `.pm`. SCEPTRE, Dia, los paquetes históricos Base/Plot y un compilador C no son requisitos para graficar una tabla con Simplot ya instalado.

Elemento	Uso
Perl y JSON::PP	Programa principal y contrato DAT.
Path::Class, Capture::Tiny	Dependencias Perl del conjunto actual; en Debian/Ubuntu, `libpath-class-perl` y `libcapture-tiny-perl`.
Gnuplot, versión 5 o posterior	Dibujo y terminales de salida; no todas las compilaciones incluyen los mismos terminales.
slex, paquete 1.0-25 o posterior	Lector `d2datdinc` con soporte gráfico actualizado.
Tabular, paquete 2.0-16 o posterior	Ejecutable `interpola`, Tabular2 y sus dependencias de unidades.
LaTeX, español, latexmk	Solamente para construir este manual, no para ejecutar Simplot.

Capítulo 3

PARAPLOT: lógica del lenguaje y forma de usarlo

3.1. Qué es PARAPLOT

3.1.1. Dos misiones y nombres de las secciones

Simplot tiene dos misiones independientes: generar una sección GNUPLOT por cada PARAPLOT y ejecutar las secciones GNUPLOT invocando Gnuplot. Por defecto hace ambas:

Pedido	Sección generada
PARAPLOT TEMPERATURAS	GNUPLOT TEMPERATURAS.
PARAPLOT sin nombre, segundo en el documento	GNUPLOT 2.
PARAPLOT ESTADOS	GNUPLOT ESTADOS.
Ningún PARAPLOT	Un único GNUPLOT 1 automático.

Puede haber varios PARAPLOT. El nombre explícito se conserva exactamente, incluidas mayúsculas y minúsculas. Sin nombre, se asigna la posición entre todos los PARAPLOT, contando desde 1 también los que tienen nombre. Cada GNUPLOT recibe el mismo identificador. Los identificadores repetidos, incluso por colisión entre un número explícito y uno automático, son errores. Los nombres de gráficos internos son locales al grupo y no deben confundirse con el nombre de la sección.

Opción	Comportamiento
Ninguna	Genera y ejecuta.
`--no-generar`	Conserva y ejecuta los GNUPLOT existentes; falla si no hay ninguno.
`--no-ejecutar`	Genera el documento sin invocar Gnuplot.
Ambas opciones	Conserva el documento sin generar ni ejecutar.
`-noX`	Suprime la ventana, no las etapas de generación o ejecución.

Opción	Comportamiento
`-save`	Guarda el documento y los SVG efectivamente producidos; por defecto el documento va a stdout.

Al regenerar se reemplazan los GNUPLOT anteriores. Estas secciones van al final del .ds; cada una incluye el programa y los datos y termina con la receta de terminal para ejecutar ese grupo mediante Gnuplot. Simplot no ejecuta esa receta mediante el shell: invoca Gnuplot directamente. Los programas GNUPLOT recibidos deben ser de confianza.

PARAPLOT es el lenguaje con el que se describe un pedido de dibujo: de qué tablas tomar los datos, qué variables poner en cada eje, qué series superponer y cómo reunir los gráficos. No es un simulador ni una tabla de resultados. El pedido puede acompañar a un modelo en un archivo .ds, pero también puede describir datos ya existentes, producidos por Calcula, Siru, SCEPTRE, Tabular, Radia, un instrumento u otro programa.

Es un lenguaje declarativo: se nombran fuentes y gráficos y se describen sus relaciones. El usuario no programa un bucle para recorrer las filas ni una instrucción por cada segmento. Simplot interpreta el pedido y utiliza Gnuplot para hacer el dibujo. Dentro de la cadena, PARAPLOT define qué dibujar en la etapa 3; Simplot con Gnuplot realiza ese pedido en la etapa 4. No son herramientas alternativas ni una clasificación de superioridad.

RESULTADOS es la sintaxis ultrasimple de este mismo lenguaje. Permite expresar el pedido indicando solamente las variables y, si hace falta, sus rótulos, magnitudes y unidades. No es un lenguaje separado ni un graficador independiente: abrevia las decisiones que la forma explícita de PARAPLOT permite detallar.

Pieza	Qué contiene y para qué sirve
Tabla	Las muestras: columnas con valores y, cuando están disponibles, nombres, magnitudes y unidades.
PARAPLOT	El pedido escrito por el usuario: fuentes, selección de coordenadas y composición.
DAT	La descripción interpretada que intercambian los programas; no contiene automáticamente todas las tablas.
Simplot con Gnuplot	Convierte el pedido y los datos en el dibujo.

La potencia del lenguaje debe estar en expresar el pedido con claridad, no en exigir al usuario conocer el JSON interno o escribir órdenes de Gnuplot. El objetivo es un PARAPLOT único, más sencillo y expresivo, compatible con las descripciones antiguas. Hoy la implementación está en transición: los ejemplos siguientes incluyen el ajuste transitorio que necesita el programa actual, y la compatibilidad pendiente se explica al final del capítulo.

3.2. Empezar por RESULTADOS: PARAPLOT ultrasimple

El primer pedido puede ser simplemente:

```
RESULTADOS
A,B
```

Es un fragmento de un documento .ds: pide dibujar A y B usando la fuente y las convenciones predeterminadas. No es preciso empezar declarando gráficos, ejes y paneles. Cuando los metadatos no bastan o se desea otra presentación, el mismo pedido se enriquece:

```
RESULTADOS
A(Interior,temperatura,tempF), B(Exterior,temperatura,tempF)
```

Grado de detalle	Qué escribe el usuario
Lista elemental	Los nombres de las variables. Es el caso más breve de RESULTADOS.
RESULTADOS con atributos	Variables y, opcionalmente, leyenda, magnitud y unidad de presentación.
PARAPLOT explícito	Fuentes, parejas X:Y, líneas o puntos, ejes y composición de gráficos.

Son formas de expresar el mismo tipo de pedido, no tres lenguajes. La abreviación se expande a fuentes, series y gráficos con la misma lógica de dibujo. En el recorrido estructurado actual, RESULTADOS toma la primera columna como abscisa, usa líneas y agrupa las ordenadas por magnitud y unidad de presentación, conservando el orden de primera aparición. La primera columna no tiene que ser tiempo. Sin fuente SIMULADO explícita, se toma la tabla con el nombre base del documento y extensión .simul.

Por ejemplo, si las columnas son T, A y B y ambas ordenadas tienen la misma magnitud y unidad, la lista A,B selecciona las parejas (T,A) y (T,B) para un gráfico común. Si sus magnitudes o unidades de presentación difieren, se organizan en grupos diferentes. Estos valores predeterminados son parte del significado de la abreviación: deben conservarse al mejorar el lenguaje y comprobarse frente al pedido explícito equivalente.

Para elegir B contra A, comparar varias fuentes o componer paneles, se utiliza la forma explícita que se explica a continuación. Esto añade control sobre el pedido; no cambia de lenguaje. La sintaxis .ds conserva los nombres históricos de las secciones RESULTADOS y PARAPLOT y sus reglas de combinación.

3.3. Los objetos: fuente, columna, serie, gráfico y panel

Concepto	Significado
Fuente	Nombre simbólico que identifica una tabla o una selección de bloque. Una ruta es la ubicación de los datos; el nombre de fuente permite referirse a ellos sin repetirla.
Columna	Variable dentro de una fuente. Se selecciona por nombre o por posición; no confundir su nombre interno con el rótulo del eje.
Serie	Una selección de coordenadas X e Y de la misma fuente, con una forma de dibujo, por ejemplo líneas o puntos. Cada fila aporta una pareja.
Gráfico	Uno o varios conjuntos de parejas dibujados sobre los mismos ejes. Superponer series no suma ni mezcla sus valores.

Concepto	Significado
Panel	Composición de varios gráficos separados. Cada gráfico conserva sus series y sus ejes.

```
Fuente propia SIMULADO (implicita)
  columnas T, A, B
        |
        +--> serie (T,A) --+
        +--> serie (T,B) --+--> grafico TEMPERATURAS --+
        |                                              +--> panel RESUMEN
        +--> serie (A,B) -----> grafico ESTADOS ------+
```

Una fuente puede alimentar varios gráficos y un gráfico puede reunir series de distintas fuentes. Cada serie conserva la correspondencia entre X e Y de su propia tabla. Comparar dos corridas no requiere que tengan la misma cantidad de filas; operar numéricamente entre ellas sí requiere preparar antes una correspondencia adecuada fuera del graficador.

3.4. Cómo se lee una orden

Una forma habitual es `ordenOBJETO=argumentos`. El verbo indica qué se declara o modifica; el objeto es el nombre de la fuente, del gráfico o del panel al que se aplica. No todas las órdenes históricas tienen exactamente esa forma: por ejemplo, `nomostrar` recibe una lista de gráficos sin signo igual.

```
lineas TEMPERATURAS = T:A,B
```

Fragmento	Lectura
`lineasTEMPERATURAS`	Añadir series de líneas al gráfico TEMPERATURAS; crearlo si todavía no existe.
Fuente omitida	Usar la única fuente disponible. Sin archivos declarados, el .ds aporta sus resultados homónimos con el nombre interno SIMULADO.
`T:`	Usar la columna T como coordenada horizontal.
`A,B`	Usar A y B como dos ordenadas: se crean las parejas (T,A) y (T,B).

El signo `:` separa la abscisa de las ordenadas; la coma enumera argumentos o columnas según su posición. No es una expresión aritmética ni un intervalo. Si las columnas físicas son T, A y B, en ese orden, `1:2,3` selecciona las mismas parejas. Los índices de columnas en estos ejemplos empiezan en 1. La columna histórica 0 significa índice de muestra y no debe confundirse con una columna medida ni con tiempo.

En el lector de estos ejemplos las palabras clave no distinguen mayúsculas, pero los identificadores y nombres de columnas sí. Para ejemplos nuevos conviene emplear nombres simples como MEDIDAS, TEMP o PRESION. El DAT admite selectores de nombres más generales que la sintaxis textual actual; eso no autoriza a inventar una forma .ds que el lector aún no reconozca. La migración deberá preservar las reglas de los documentos históricos, no imponerles retrospectivamente las nuevas.

3.5. Un pedido completo, explicado por decisiones

Este es el ejemplo ejecutable `ejemplos/comparacion.ds`; usa su tabla homónima `ejemplos/comparacion.simul`. No se declara `archivo` ni se repite el nombre de la fuente porque solo hay una. No necesita resolver un circuito.

```
PARAPLOT
lineas TEMPERATURAS = T:A,B
titulo TEMPERATURAS = Interior y exterior
ejes TEMPERATURAS = Tiempo(s),Temperatura(tempC)
puntos ESTADOS = A:B
titulo ESTADOS = Exterior contra interior
ejes ESTADOS = Interior(tempC),Exterior(tempC)
mostrarjuntos RESUMEN = TEMPERATURAS,ESTADOS
```

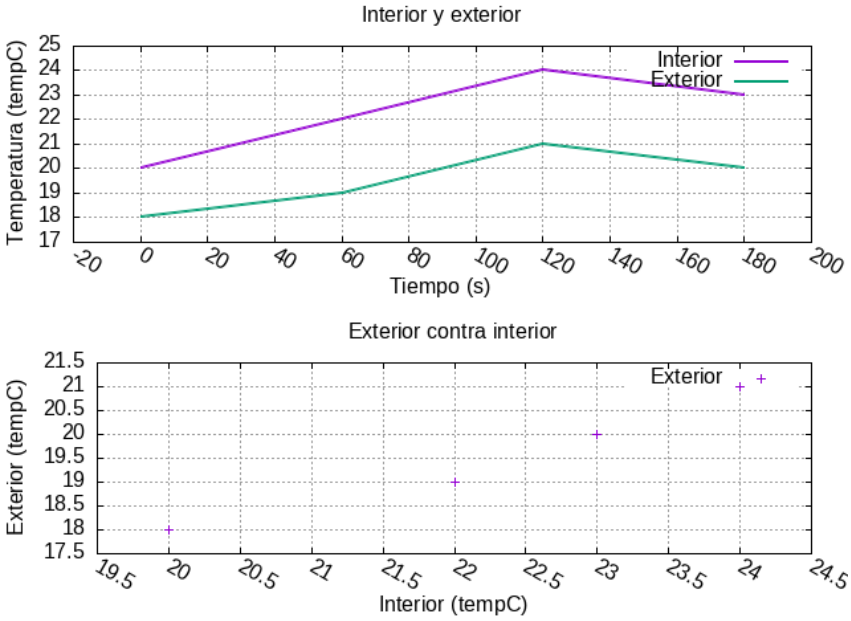


Figura 3.1: Gnuplot: salida del pedido completo con fuente propia implícita.

1. `PARAPLOT` abre la sección de pedidos gráficos, que termina al comenzar otra sección o al llegar al fin del archivo. No se escribe una orden de cierre.
2. El nombre del documento determina la tabla propia: comparacion.ds usa comparacion.simul. El lector completa internamente la fuente SIMULADO.
3. `lineas` pide dos evoluciones contra T, sobre los mismos ejes.
4. `titulo` da un texto al gráfico; `ejes` fija rótulos y unidades de presentación.
5. `puntosESTADOS=A:B` cambia las coordenadas: ahora A es horizontal y B vertical. T no interviene en ese gráfico.
6. `mostrarjuntos` reúne ambos gráficos en un panel. No superpone sus coordenadas.

En el capítulo siguiente se muestra cómo ejecutarlo y la figura que produce. El lector del modelo estructurado resuelve referencias adelantadas: no es un guion que dibuje inmediatamente al leer cada línea. Para facilitar la lectura humana conviene escribir primero las fuentes, después las series y los atributos, y finalmente los paneles.

3.6. Superponer no es poner en paneles

```
lineas COMPARACION = T:A
puntos COMPARACION = T:B
mostrarjuntos RESUMEN = COMPARACION,ESTADOS
```

Las dos primeras órdenes tienen el mismo destino y agregan series al mismo gráfico; no reemplazan una por otra. Una se dibuja con líneas y la otra con puntos. La última orden compone dos gráficos distintos. En el modelo actual las series se conservan en orden de incorporación; los atributos como título o rango configuran el objeto nombrado.

Intención	Cómo expresarla
Varias Y contra la misma X	`lineasG=X:Y,Z`; dos series en G, con una fuente única.
Mezclar líneas y puntos	Repetir órdenes de distinto tipo sobre el mismo gráfico G.
Comparar archivos	Declarar fuentes diferentes y añadir sus series al mismo gráfico.
Ver gráficos separados juntos	`mostrarjuntosP=G,H`; un panel con G y H.
No publicar cada gráfico por separado	Existe `nomostrar` en el lenguaje histórico; su incorporación al recorrido común está pendiente.

Actualmente `mostrarjuntos` en los ejemplos .ds de este capítulo produce un panel vertical, y se generan también los gráficos individuales. No admite paneles anidados ni un mismo gráfico repetido dentro del panel. El número de columnas puede configurarse en DAT, pero aún falta su forma textual PARAPLOT.

3.7. Varias fuentes: nombrar la propia implícita y la externa

En un documento con resultados propios no hace falta repetir su nombre de archivo. Pero cuando se comparan esos resultados con otra tabla, cada serie debe indicar de cuál toma sus valores. La fuente propia tiene el nombre SIMULADO aunque no se haya declarado con `archivo`.

```
PARAPLOT
archivo MEDICIONES = mediciones.simul
lineas COMPARACION_DATOS = SIMULADO,T:TEMP
puntos COMPARACION_DATOS = MEDICIONES,T:TEMP
titulo COMPARACION_DATOS = Calculo y mediciones
ejes COMPARACION_DATOS = Tiempo(s),Temperatura(tempC)
```

Referencia	De dónde toma los valores
`SIMULADO`	Fuente implícita: dos-fuentes.simul, por el nombre de dos-fuentes.ds. No hay una orden archivo para ella.
`MEDICIONES`	Fuente externa: mediciones.simul, declarada expresamente. Su nombre no se deduce del documento.
`COMPARACION_DATOS`	Un solo gráfico: líneas del cálculo y puntos de mediciones sobre los mismos ejes.

Las dos tablas pueden tener distinta cantidad de muestras y distintas abscisas. Cada serie toma

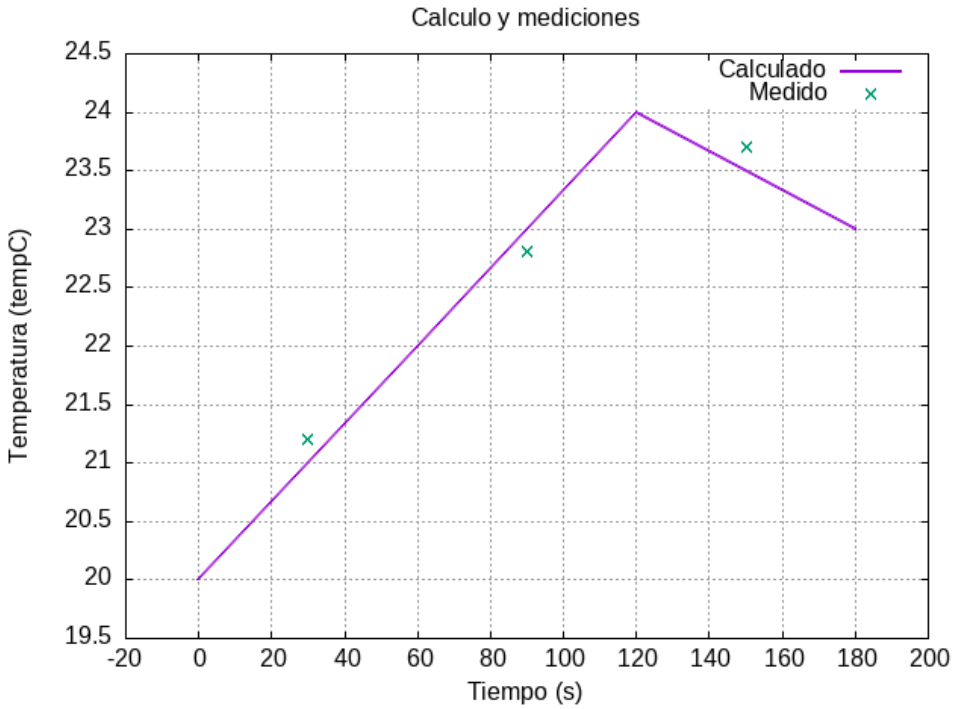


Figura 3.2: Gnuplot: líneas de SIMULADO y puntos de MEDICIONES en un mismo gráfico.

X e Y de su propia tabla: no se suman filas ni se interpola para hacerlas coincidir. El capítulo de uso muestra las tablas del ejemplo, el comando y el dibujo real.

Cantidad de fuentes	Regla
Una, implícita	`lineasG=T:TEMP`; se completa SIMULADO y la tabla homónima del .ds.
Una, declarada	También puede omitirse el nombre; se completa el de la única fuente declarada. No se agrega otra fuente propia que nadie pidió.
Dos o más	`lineasG=SIMULADO,T:TEMP` y `puntosG=MEDICIONES,T:TEMP`. Si se omite la fuente, se rechaza la ambigüedad.
Forma antigua explícita	Se sigue aceptando `FUENTE,X:Y` aunque solo haya una fuente. Una declaración explícita de SIMULADO prevalece sobre el nombre homónimo.

La abreviación se resuelve después de reunir todas las declaraciones y referencias; no depende de que la tabla externa se declare antes o después de la serie. Un nombre desconocido no se convierte silenciosamente en SIMULADO. Este soporte está comprobado en .ds y .ds2; el lector histórico de .d todavía no propaga el selector técnico del modelo estructurado, una limitación pendiente distinta de la regla del lenguaje.

3.8. PARAPLOT independiente: declarar dónde están los datos

Un .ds puede contener solamente la sección PARAPLOT, sin describir un circuito. Si la tabla externa no es homónima, la declara. El ejemplo ejecutable `ejemplos/independiente.ds` usa

índices de columnas:

```
PARAPLOT
archivo MEDIDAS = medidas.simul
lineas TEMPERATURAS = MEDIDAS,1:2,3
```

```
simplot -graf=4 -sinventana independiente.ds
```

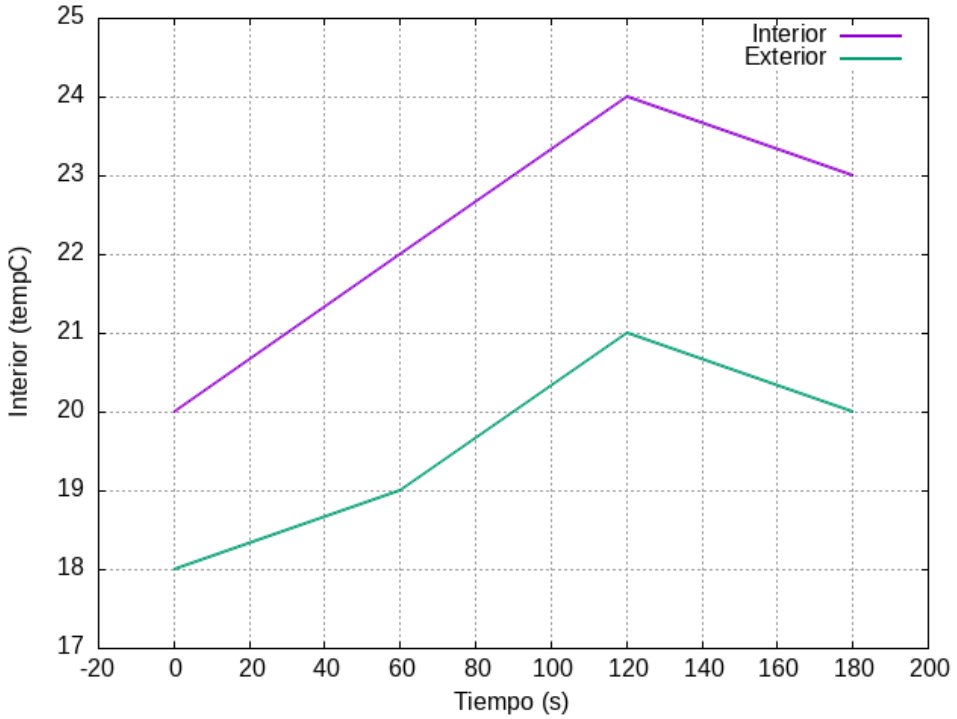


Figura 3.3: Gnuplot: salida del PARAPLOT independiente con fuente declarada.

MEDIDAS es el nombre simbólico y medidas.simul es la ruta, relativa al documento gráfico. La forma explícita conserva compatibilidad con los PARAPLOT independientes antiguos. La misma orden sirve dentro de un .ds para agregar tablas externas, como MEDICIONES en el ejemplo anterior; no se necesita para volver a declarar sus propios resultados.

3.9. Ejes, rótulos, unidades y encuadre

```
titulo TEMPERATURAS = Interior y exterior
ejes TEMPERATURAS = Tiempo(s),Temperatura(tempF)
rangox TEMPERATURAS = 0,10
rangoy TEMPERATURAS = 50,100
```

Pedido	Efecto
Título	Describe el gráfico; no renombra columnas ni crea datos.
Rótulo del eje	Texto que explica la coordenada; no demuestra cuál era la unidad de origen.
Unidad entre paréntesis	Pide presentar los valores en esa unidad. Tabular2 realiza la conversión a partir de la unidad de origen conocida.

Pedido	Efecto
Rango	Fija la región visible del eje. No elimina filas de la fuente ni equivale a `selec`. Se expresa en las coordenadas de presentación.

Si la tabla contiene temperaturas en `tempC` y el eje pide `tempF`, se convierten los valores, no solo el texto. Para temperaturas absolutas se usan `tempC/tempF`; para intervalos, `degC/degF`. Una unidad desconocida o incompatible requiere corregir el dato o la declaración; no se adivina.

En la sintaxis de estos ejemplos los rangos contienen dos números finitos separados por coma y con mínimo menor que máximo. La forma histórica usa dos puntos; esa diferencia actual es parte del trabajo de compatibilidad. No se debe cambiar la interpretación de un PARAPLOT antiguo por aceptar una forma nueva.

3.10. Un punto, una trayectoria y los límites de error

Una fila de una tabla representa una muestra o un estado. PARAPLOT no exige que haya tiempo ni muchas filas: `puntosESTADOS=A:B` ubica el estado por sus coordenadas A y B cuando hay una sola fuente. Una tabla de estados psicrométricos puede dibujarse de esta manera. Las tablas de curvas que se quieran añadir al diagrama son otras series proporcionadas al graficador.

Con varias filas, `lineas` conecta las muestras en el orden recibido. El pedido no ordena ni remuestrea automáticamente los datos. Si hacen falta otros puntos por interpolación, Tabular2 puede producirlos antes; PARAPLOT describe cómo dibujar el resultado.

Las barras de error ya admiten límites absolutos aportados en columnas:

```
puntos G = DATOS,X:Y,limites_y=YMIN:YMAX
puntos H = DATOS,X:Y,limites_x=XMIN:XMAX,limites_y=YMIN:YMAX
```

Cada fila aporta centro y extremos. Estos límites no son deltas ni porcentajes, y Simplot no estima incertidumbres. Con límites se escribe una pareja X:Y por orden; para superponer varias series se repite la orden sobre el mismo gráfico. El capítulo de uso desarrolla validación, unidades y un ejemplo completo del corpus.

3.11. El mismo pedido en .ds, DAT y el pipe

PARAPLOT es una sección opcional del .ds, no un tipo de archivo separado. El contrato común contempla un solo .ds con una o varias secciones TABLE, con o sin PARAPLOT; un .ds acompañado por una o más tablas externas; o el documento textual completo por entrada estándar. La tabla puede preceder o seguir al pedido. Una única TABLE no necesita nombre; varias usan nombres lógicos. No se requiere DOCUMENT ni END.

Slex conserva TABLE en el DAT y vincula sus nombres con las fuentes gráficas. Simplot carga las tablas mediante Tabular2. Funcionan los archivos separados y la entrada estándar textual; el corpus `corpus/entradas-ds/` y `tests/documento.t` comprueban también grupos, ejecución de GNUPLOT existente y SVG reales. La traducción de todos los pedidos históricos a la interfaz documental todavía no está terminada; sus opciones de compatibilidad siguen disponibles.

Siru identifica la tabla de NOMBRE.ds como `TABLENOMBREFORMATO=SIMUL` al incorporarla al documento de salida. El pipe conserva ese nombre aunque el archivo numérico tenga otro nombre.

La referencia implícita SIMULADO de los pedidos anteriores se mantiene por compatibilidad.

Entrada	Papel de PARAPLOT
Tabla sin instrucciones	Simplot puede elegir gráficos automáticos sin que el usuario escriba PARAPLOT.
RESULTADOS	Sintaxis ultrasimple de PARAPLOT: variables y atributos opcionales; las convenciones completan fuentes, series y gráficos.
PARAPLOT dentro de .ds	Especifica fuentes, coordenadas y composición explícitamente. El mismo lector slex sirve a Siru y a Simplot.
.ds solo con PARAPLOT	Contiene las órdenes gráficas sin requerir el resto del modelo del sistema.
DAT	Conserva el pedido interpretado para dibujarlo sin volver a leer el .ds; las tablas siguen siendo necesarias.
Pipe desde Siru	Transporta la descripción interpretada y la tabla a Simplot; no requiere un archivo de intercambio entre ambos procesos.

En la sintaxis histórica de secciones, la sección llamada PARAPLOT reemplaza normalmente los gráficos abreviados en RESULTADOS. Para conservarlos y agregar los explícitos se usa `ANADIR_GRAFICOS=1` o la sección `GRAFICOS_ADICIONALES`. Es una regla de combinación de dos formas del mismo lenguaje, no de dos lenguajes independientes. No basta con declarar una fuente para expresar que se desean todos los gráficos abreviados además de los nuevos.

`simplot--stdin` recibe el documento textual .ds, con tablas y PARAPLOT opcionales; Siru usa ese mismo transporte. El protocolo JSON anterior se conserva solo por compatibilidad. Siru llama a Simplot por defecto; `-nosimplot` anula esa llamada. La receta final de cada GNUPLOT es para Bash: usa `gnuplot-persist<(awk...)` para extraer y ejecutar ese grupo. El nombre de documento de la receta se fija con `--salida`; si se redirige stdout, debe usarse ese nombre.

3.12. Compatibilidad: qué existe hoy y qué debe conservarse

PARAPLOT es un solo lenguaje. La implementación todavía conserva un recorrido de compatibilidad y otro basado en un modelo estructurado. El lector elige automáticamente el recorrido estructurado para las órdenes que ya admite y conserva el histórico para las demás. La unificación todavía no está terminada: no se debe afirmar que el recorrido común ya cubre toda entrada antigua.

Familia	Situación actual y objetivo
Fuentes, líneas, puntos, títulos, ejes y paneles	Cubiertas en los ejemplos del modelo estructurado; conservar significado, referencias y combinación con RESULTADOS.
`variables`, `magnitudes`, `unidades`	Metadatos históricos asociados a fuentes. Completar su traducción, distinguiéndolos de la sección UNIDADES y de los rótulos de presentación.
`selec`	Selección histórica de filas; conservar límites y dependencias. El procesamiento se integra mediante Tabular2, no con otro algoritmo numérico dentro del nuevo motor.
`arch_var`, `variaciones`	Familias de tablas y rótulos de corridas. Mantener identidad de cada corrida y no confundirlas con bloques de un archivo.

Familia	Situación actual y objetivo
`funcion`	Referencia analítica histórica. Preservar la prestación mediante un contrato de expresiones y el componente numérico adecuado, sin admitir ejecución arbitraria.
`nomostrar`, `guardar`	Control histórico de presentación y conservación del guion; migrar su significado explícitamente.
Bloques, escalas, formatos y disposición avanzada	Ya hay capacidades en DAT sin forma .ds. Todas deben poder pedirse en PARAPLOT.

La condición de llegada es que los PARAPLOT antiguos válidos se sigan leyendo sin reescritura obligatoria y que las formas nuevas se traduzcan al mismo modelo y al mismo motor. No se eliminará el recorrido histórico antes de probar la equivalencia. Los documentos originales del corpus deben conservarse: sustituirlos por versiones nuevas ocultaría incompatibilidades.

3.13. Cómo será más claro, potente y expresivo

Lo que sigue es diseño acordado, no sintaxis nueva ya disponible. No se publican aquí órdenes ficticias que el lector rechazaría.

Principio	Consecuencia para quien escribe PARAPLOT
Casos simples breves	Elegir X e Y y una fuente debe seguir requiriendo pocas palabras, sin JSON ni programación gráfica.
Opciones explícitas	Las extensiones complejas tendrán nombres reconocibles; evitar largas listas de argumentos cuyo significado dependa solo de la posición.
Composición	Combinar series, estilos, escalas y paneles en lugar de inventar una orden diferente para cada combinación.
Compatibilidad	Mantener las formas antiguas como entradas válidas; la simplificación nueva es una posibilidad, no una obligación.
Diagnóstico	Indicar qué orden, objeto, columna o unidad impide dibujar; no omitir silenciosamente lo que no se entienda.

De Observable Plot interesan las ideas que enriquecen el pedido de dibujo: capas combinables, asociación de propiedades visuales con columnas, leyendas coherentes y facetas. No hace falta adoptar JavaScript ni agregar Observable como dependencia.[3, 7]

Ampliación prevista	Qué aportará a PARAPLOT
Capas gráficas	Combinar líneas, puntos, barras, escalones, bandas entre límites, segmentos y textos sobre ejes compartidos. Las barras de error ya están disponibles.
Propiedades visuales	Declarar color, tamaño y símbolo mediante constantes o columnas suministradas; separar estos atributos de las coordenadas.
Escalas y leyendas	Describir cómo se leen coordenadas y atributos, con unidades y leyendas explícitas.

Ampliación prevista	Qué aportará a PARAPLOT
Facetas	Repetir un gráfico por categorías de datos ya recibidos; no confundirlo con superponer series ni con calcular un resumen estadístico.

El procesamiento numérico general permanece en Tabular2 u otros productores. Dibujar barras de frecuencias no implica calcular el histograma dentro de Simplot; dibujar una banda no implica estimar sus cotas. Los requisitos de categorías y texto se coordinan con Tabular2. El plan detallado y sus criterios de aceptación están en `doc/PLAN-FUNCIONAL.md`, sección «PARAPLOT único».

Capítulo 4

Uso concreto: de una tabla a una presentación

4.1. Preparar un lugar de trabajo

Siru llama a Simplot al terminar y transmite por un pipe las secciones textuales TABLE y PARAPLOT con todos los datos calculados. `siru-nosimplotcircuito.ds` omite esa llamada. El receptor usa `simplot--stdin`: no necesita abrir el .ds ni la tabla de Siru. El nombre de la tabla es el del caso, sin ruta ni extensión, aunque el archivo de resultados tenga un sufijo. El protocolo JSON anterior se conserva solo como compatibilidad. El contrato está en `doc/DAT-PARAPLOT.md` y los casos ejecutables en `corpus/entradas-ds/`.

Los motores actuales preparan copias internas de lo recibido en `simplot-entrada-XXXXXX/`; no son archivos compartidos con Siru. Las tablas externas de comparación siguen resolviéndose respecto del documento original. Esta modalidad dibuja al terminar, sin diálogo interactivo ni animación en vivo. La entrada habitual por archivos se mantiene. Para no mezclar resultados con el corpus versionado, copiar los ejemplos a un directorio propio. En las recetas siguientes, `simplot` es el programa instalado. Para usar las fuentes, sustituirlo por `perl-I/ruta/simplot/lib/ruta/simplot/scripts/simplot.pl`.

```
mkdir -p /tmp/mi-simplot
cp doc/manual/ejemplos/* /tmp/mi-simplot/
cd /tmp/mi-simplot
```

No es obligatorio un temporal: puede ser un directorio persistente de trabajo. Se recomienda separar entradas, resultados y archivos versionados. Algunos caminos de la CLI histórica escriben registros en el directorio actual; seleccionar un directorio de salida no implica que absolutamente todos los registros cambien de lugar.

4.2. Sin instrucciones: una tabla numérica

```
# para Siru: sceptrevars variables magnitudes unidades
# T A B
# Tiempo,Interior,Exterior
# tiempo,temperatura,temperatura
# s,tempC,tempC
0 20 18
60 22 19
```

```
120 24 21
180 23 20
```

La cabecera SIMUL de este ejemplo enumera nombres internos, rótulos, magnitudes y unidades. Las filas son muestras, con separación por espacios. Con una tabla sencilla también se admiten cabeceras más simples, pero entonces no se debe inventar información de unidades que no contiene el archivo.

```
simplot -graf=4 -sinventana medidas.simul
```

Sin descripción, el camino histórico dibuja las columnas posteriores contra la primera y puede agruparlas según sus metadatos. Aquí es tiempo porque la tabla lo dice; una tabla diferente puede tener presión en su primera columna. `-graf=4` selecciona producción no interactiva, no significa "no dibujar". Para CSV y fechas se recomienda el ejemplo descrito más adelante, que usa explícitamente Tabular2.

En esta tabla el modo automático agrupa Interior y Exterior por sus metadatos de temperatura y presenta el tiempo en horas según su configuración histórica.

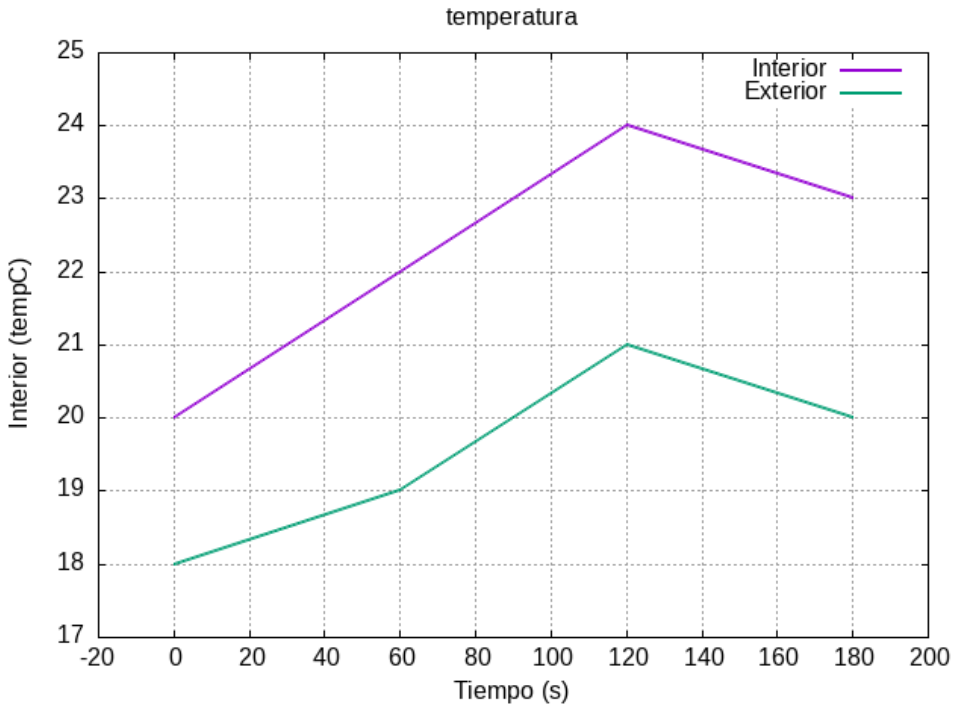


Figura 4.1: Gnuplot: salida automática real de medidas.simul.

4.3. Elegir columnas de forma interactiva

```
simplot -i -sinventana medidas.simul
```

El diálogo también funciona sobre documentos .ds, .d y .dat mediante el renderizador común. Una respuesta `1:2` elige la columna 1 como X y la 2 como Y; `1:2,3` superpone dos ordenadas. Se admiten nombres de columna o leyendas, por ejemplo `'Tiempo':'Temperatura'`. La columna especial 0, solo como X en este diálogo, es el índice de muestra de Gnuplot: no es una medición de tiempo. Una respuesta vacía o fin de entrada termina la selección. Si hay varias fuentes se eligen

primero por sus números. Después de pedir dos o más gráficos, se pueden reunir en paneles o superponer sus series; las unidades de un mismo eje deben ser compatibles. La selección produce el mismo modelo que PARAPLOT, sin otro lector de tablas ni otro motor de dibujo. Con `-i`, los pedidos gráficos del documento se sustituyen por los del diálogo.

Para reproducir esta salida, responder `1:2,3` y después una línea vacía:

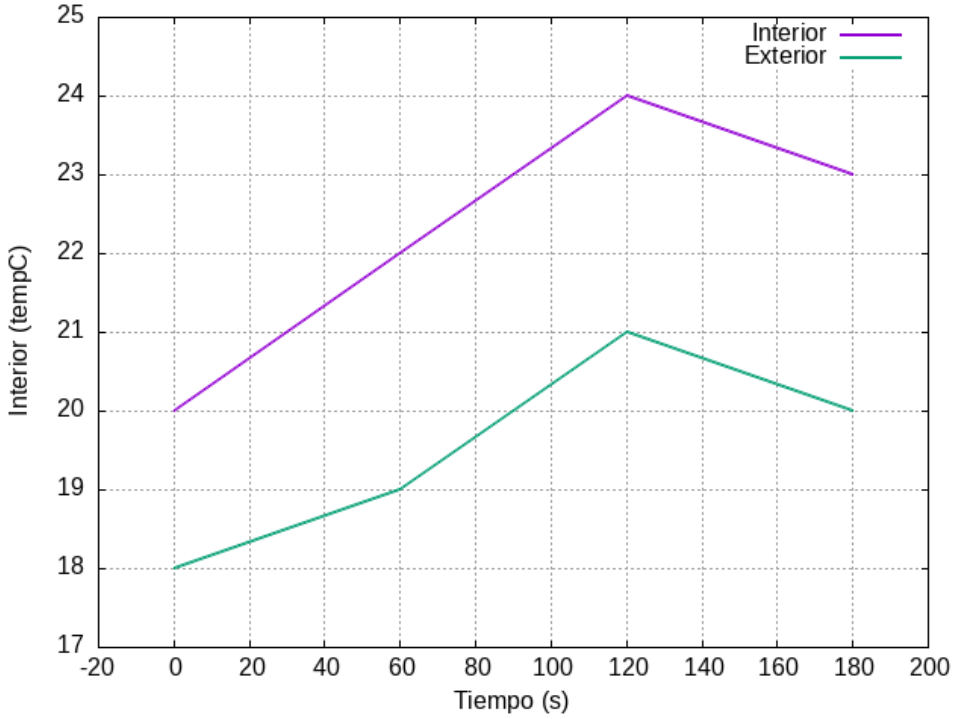


Figura 4.2: Gnuplot: selección interactiva de las columnas 2 y 3 contra la 1.

4.4. PARAPLOT: vista temporal y espacio de estados

```
PARAPLOT
lineas TEMPERATURAS = T:A,B
titulo TEMPERATURAS = Interior y exterior
ejes TEMPERATURAS = Tiempo(s),Temperatura(tempC)
puntos ESTADOS = A:B
titulo ESTADOS = Exterior contra interior
ejes ESTADOS = Interior(tempC),Exterior(tempC)
mostrarjuntos RESUMEN = TEMPERATURAS,ESTADOS
```

```
simplot -graf=4 -sinventana comparacion.ds
```

Orden del ejemplo	Interpretación
Fuente sin declarar	comparacion.ds usa comparacion.simul; SIMULADO se completa internamente.
`lineasTEMPERATURAS=T:A,B`	Dos series con la misma abscisa, en el mismo gráfico.
`puntosESTADOS=A:B`	B contra A; el tiempo ya no es una coordenada.

Orden del ejemplo	Interpretación
`ejes...`	Rótulos y unidades de presentación.
`mostrarjuntosRESUMEN=...`	Panel con ambos gráficos, sin fusionar sus ejes.

Este documento genera dos gráficos individuales y un panel. Al no seleccionar formatos explícitos en un DAT, el renderizador del modelo estructurado produce PNG y EPS: seis imágenes en total. La selección de formatos mediante DAT se muestra en el ejemplo de bloques. Las palabras de las órdenes no dependen de mayúsculas; los identificadores del ejemplo y los nombres de columnas sí deben respetarse.

4.5. Superponer dos fuentes: cálculo propio y mediciones

El ejemplo `dos-fuentes.ds`, explicado en la sección 3.7, declara únicamente la fuente externa MEDICIONES. SIMULADO se resuelve por el nombre del documento. Estos son datos didácticos para comprobar el dibujo, no resultados de una medición real.

Tabla	Contenido
`dos-fuentes.simul`	Cuatro muestras del cálculo: fuente propia implícita SIMULADO.
`mediciones.simul`	Tres muestras en instantes intermedios: fuente externa MEDICIONES.

```
# para Siru: sceptrevars variables magnitudes unidades
# T TEMP
# Tiempo,Calculado
# tiempo,temperatura
# s,tempC
0 20
60 22
120 24
180 23
```

```
# para Siru: sceptrevars variables magnitudes unidades
# T TEMP
# Tiempo,Medido
# tiempo,temperatura
# s,tempC
30 21.2
90 22.8
150 23.7
```

```
simplot -graf=4 -sinventana dos-fuentes.ds
```

Las líneas y los puntos pertenecen al mismo gráfico porque ambas órdenes nombran `COMPARACION_DATOS`. No tienen que coincidir las filas ni los instantes. Las dos series conservan sus valores y unidades.

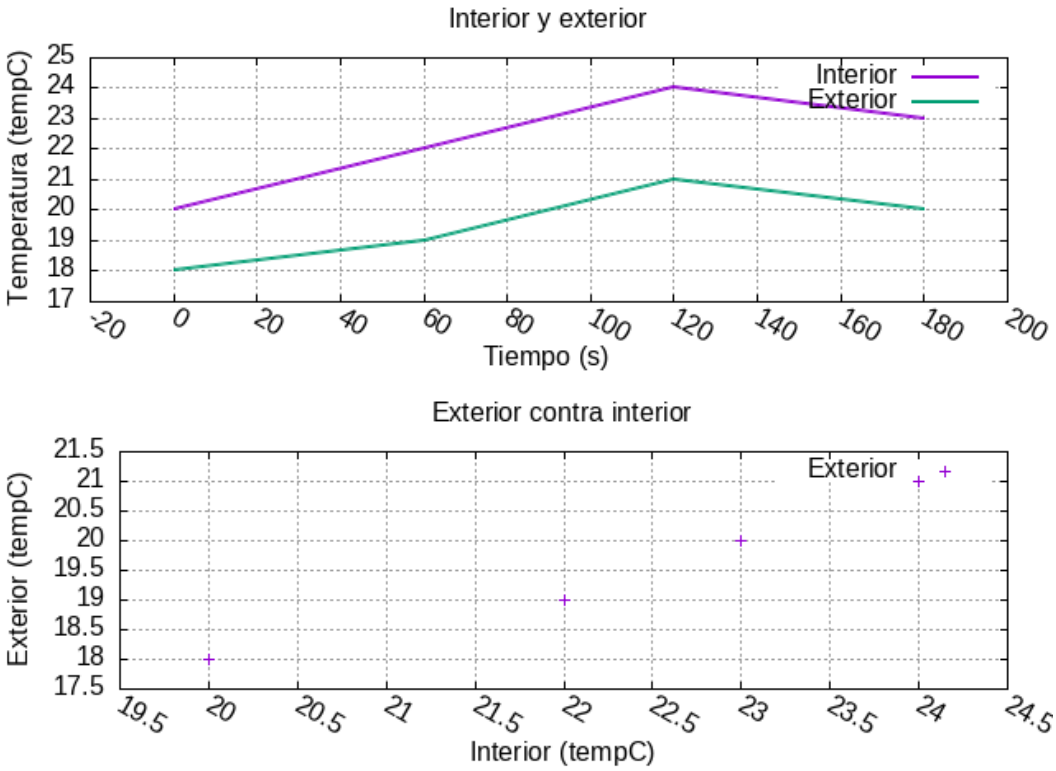


Figura 4.3: Salida real de comparacion.ds: evolución temporal y proyección B contra A.

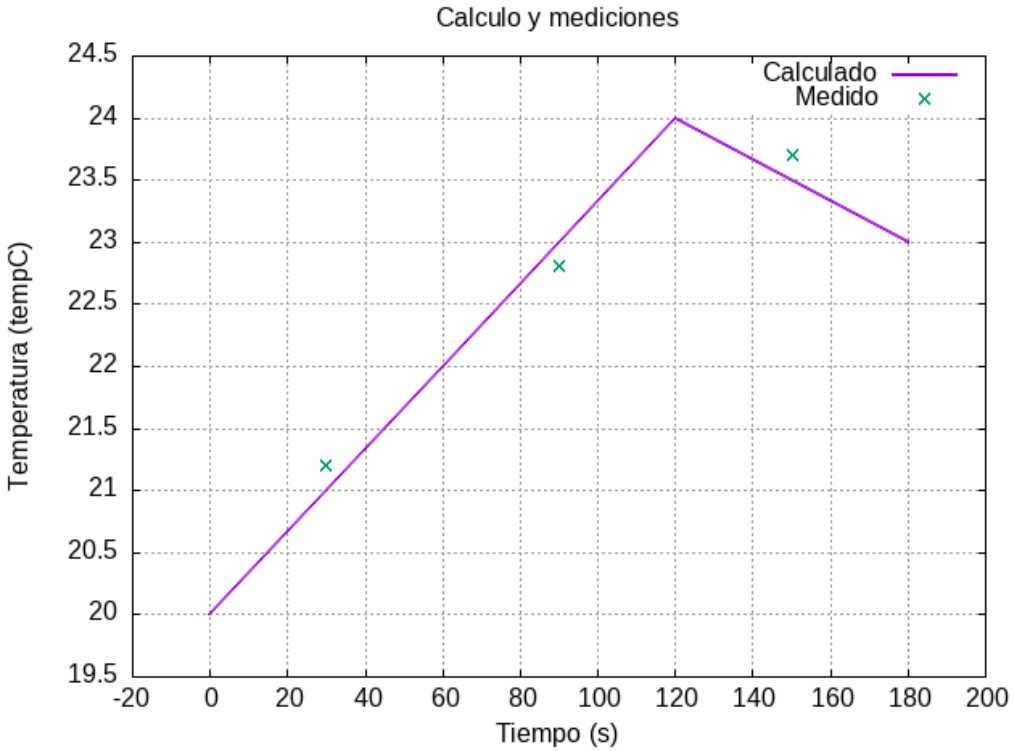


Figura 4.4: Fuente propia SIMULADO y fuente externa MEDICIONES superpuestas; datos didácticos.

4.6. PARAPLOT: barras de error con límites recibidos

Simplot dibuja los límites que le entrega el productor; no calcula incertidumbres ni estadísticas. La tabla contiene el centro y los extremos absolutos de cada intervalo, en la misma fila. Esta ampliación requiere las fuentes de Simplot y slex desde r2760; no está en los paquetes anteriores.

```
# para Siru: sceptrevars variables magnitudes unidades
# T TEMP TMIN TMAX TINF TSUP
# Tiempo,Temperatura,Tiempo inferior,Tiempo superior,Temperatura inferior,Temperatura
    superior
# tiempo,temperatura,tiempo,tiempo,temperatura,temperatura
# s,tempC,s,s,tempC,tempC
1 20 0.8 1.2 18 23
2 22 1.7 2.4 21 25
3 24 2.9 3.1 22 26
```

```
PARAPLOT
puntos VERTICAL = T:TEMP,limites_y=TINF:TSUP
puntos AMBOS = T:TEMP,limites_x=TMIN:TMAX,limites_y=TINF:TSUP
lineas UNIDOS = T:TEMP,limites_y=TINF:TSUP
ejes AMBOS = Tiempo(s),Temperatura(tempF)
mostrarjuntos BARRAS = VERTICAL,AMBOS,UNIDOS
```

```
simplot -graf=4 -sinventana barras.ds
```

Pedido	Interpretación
`limites_y=TINF:TSUP`	Barras verticales entre esas columnas.
`limites_x=TMIN:TMAX`	Barras horizontales; se pueden combinar con las verticales.
`puntos`	Centro con sus barras, sin unir las muestras.
`lineas`	Une los centros conservando las barras; una sola muestra sigue siendo visible.
`mostrarjuntos`	Compone gráficos con barras igual que otros gráficos.

Cada orden con límites describe una sola pareja X:Y; para superponer series se repiten órdenes sobre el mismo gráfico. Los selectores admiten nombres o índices. Los límites pueden ser asimétricos y deben cumplir inferior $\leq$ centro $\leq$ superior; se rechazan columnas ausentes, datos no finitos, unidades incompatibles y extremos no positivos en ejes logarítmicos.

Los extremos son valores absolutos, no deltas ni porcentajes. Una temperatura y sus límites usan `tempC`/`tempF`, no `degC`/`degF`. Por ejemplo, el centro 20 y los extremos 18 y 23 en `tempC` pasan mediante Tabular2 a 68, 64,4 y 73,4 en `tempF`. Simplot no aplica factores propios. Sin unidad de destino explícita usa la del centro; si no hay unidades declaradas en ninguna columna, admite números sin unidad.

El corpus `corpus/barras-error/errores.ds` y su tabla contienen un ejemplo completo. `test s/barras-error.t` ejecuta el lector y el dibujo real, verifica las columnas enviadas a Gnuplot, unidades, PNG/EPS/PDF/SVG, punto aislado, DAT independiente e ingreso por stdin. Las entradas no se modifican.

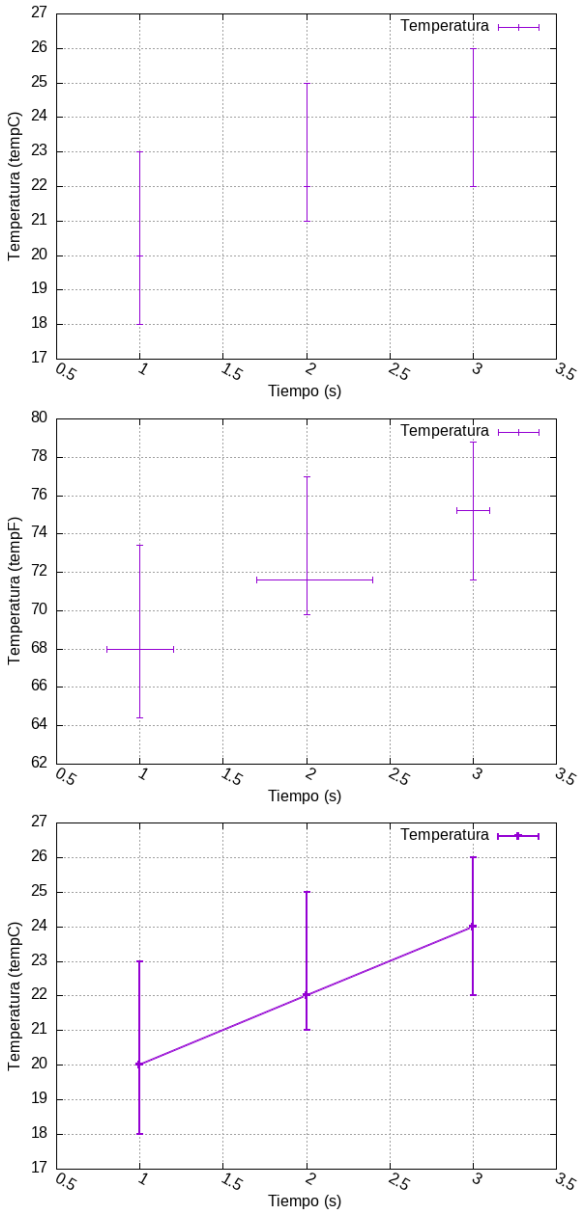


Figura 4.5: Gnuplot: límites verticales, límites en ambos ejes y centros unidos. El panel central convierte a tempF.

4.7. RESULTADOS: la sintaxis ultrasimple de PARAPLOT

Se usa el mismo lenguaje descrito en el capítulo 3, en su forma abreviada: se indican las variables y las convenciones completan el pedido gráfico. La forma explícita de PARAPLOT permite precisar las decisiones que aquí se dejan predeterminadas.

```
CIRCUIT_DESCRIPTION
Unidades de origen y unidades de presentacion distintas.
RUN_CONTROLS
RESULTADOS
A(Interior,temperatura,tempF), B(Exterior,temperatura,tempF)
UNIDADES
plantilla=Generico
A:tempC,temperatura
B:tempC,temperatura
```

```
cp medidas.simul unidades.simul
simplot -graf=4 -sinventana unidades.ds
```

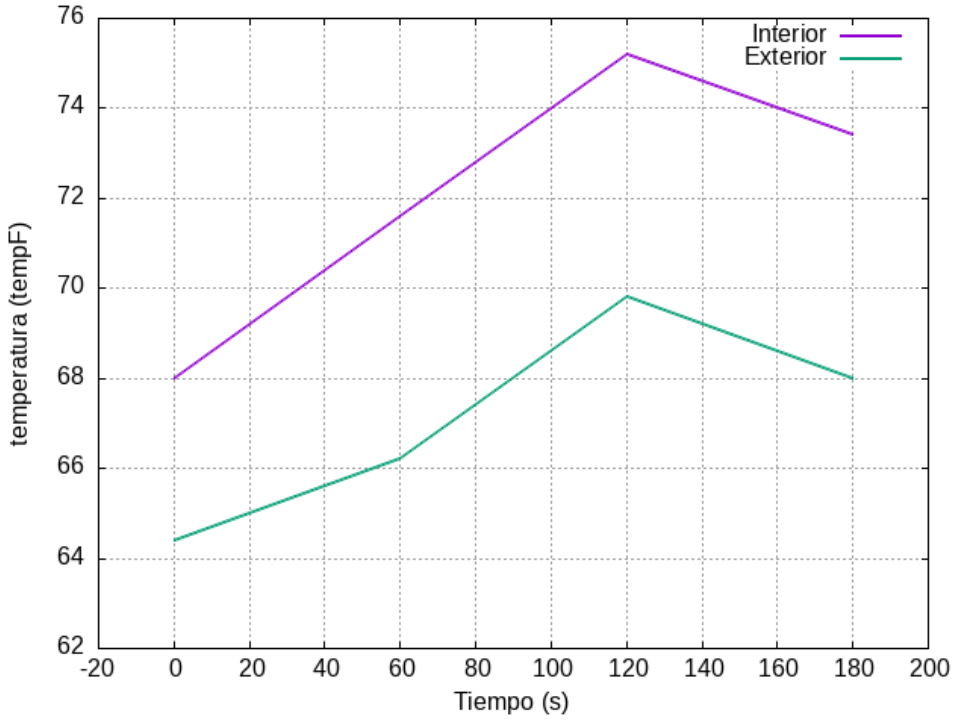


Figura 4.6: Gnuplot: RESULTADOS abreviado con temperaturas presentadas en tempF.

La fuente predeterminada se llama como el documento con extensión `.simul`; de ahí la copia explícita. RESULTADOS pide variables con sus rótulos, magnitudes y unidades de presentación. UNIDADES declara reglas de origen cuando esa información no está en la tabla. En este caso el origen ya figura como `tempC` en la cabecera: el resultado pide `tempF`. Las comprobaciones verifican que 20 pasa a 68, no solamente que el eje diga Fahrenheit.

Los resultados se agrupan por magnitud y unidad de presentación, conservando el orden de aparición. Cambiar una leyenda no calcula una variable. Una lista elemental selecciona columnas sin requerir la descripción más rica de cada resultado. En el modelo producido desde OUTPUTS, la primera variable solicitada puede actuar como X; también se admite la selección explícita `X:Y,Z`. No se agrega tiempo inexistente.

Combinación	Regla
Solo RESULTADOS	Se expande el pedido abreviado a gráficos.
GRAFICOS o PARAPLOT	Reemplaza la descripción automática de resultados, salvo control explícito de adición.
`ANADIR_GRAFICOS=1`	Conserva los resultados y agrega los gráficos indicados.
`GRAFICOS_ADICIONALES`	Expresa la intención de agregar, no reemplazar.

Estas son las reglas actuales de las secciones abreviada y explícita del mismo lenguaje. No introducir una sección PARAPLOT con solamente una declaración de archivo suponiendo que conservará automáticamente todos los gráficos de RESULTADOS. La intención de combinar debe quedar explícita.

4.8. Unidades: origen, presentación y temperatura

Concepto	Regla práctica
Unidad de origen	Primero la tabla; UNIDADES sirve de respaldo cuando falta. Una unidad pedida para mostrar no demuestra cuál era el origen.
Unidad de presentación	Una elección explícita de RESULTADOS o del eje tiene prioridad sobre la presentación deducida de los metadatos.
Unidad desconocida	No se convierte por adivinación. Corregir los metadatos o proporcionar el origen.
Series sobre un mismo eje	Deben llegar a una unidad común; indicar una unidad destino evita mezclar unidades distintas.
Magnitud	Describe el significado físico y ayuda a agrupar; no reemplaza la definición de la unidad.

`tempC` y `tempF` representan temperaturas absolutas mediante conversiones con desplazamiento. `degC` y `degF` representan intervalos; `K` se usa para kelvin. Un valor de 20 `tempC` corresponde a 68 `tempF`; una diferencia de 20 `degC` no corresponde a esa temperatura absoluta. No usar `C` ni `F` como abreviaturas de temperatura: tienen otros significados en el sistema de unidades. En expresiones compuestas, como una conductancia térmica, usar las unidades de intervalo apropiadas, no funciones de temperatura absoluta.

Las reglas de UNIDADES reconocen prefijos y admiten alias dentro de su propio contrato; no son un diccionario libre que haga válidos todos los nombres antiguos. Debe declararse la plantilla correspondiente, como `Generico`. Las coincidencias ambiguas deben corregirse, no resolverse por azar. No se supone implícitamente que todo problema sea térmico.

Tabular2 puede normalizar a SI al leer determinados formatos sencillos de cabecera. En el formato SIMUL enriquecido se conserva información de origen para las conversiones de presentación. Por eso conviene inspeccionar la tabla exportada antes de concluir que un valor se perdió: por ejemplo, una temperatura expresada en kelvin puede ser la misma magnitud que estaba en Celsius en otro formato. Véanse también `doc/manual-extraido/unidades.tex` y la documentación de Tabular2 para el contexto histórico y el contrato actual, respectivamente.

4.9. Un solo punto: graficar el espacio de coordenadas

```
# para Siru: sceptrevars variables magnitudes unidades
# TEMP TBH WWW
# Temperatura,Bulbo humedo,Humedad absoluta
# temperatura,temperatura,humedad
# tempC,tempC,1
25 18 0.010
```

```
simplot --pares --columnas TEMP,TBH,WWW \
  --formatos png --directorio pares punto.simul
```

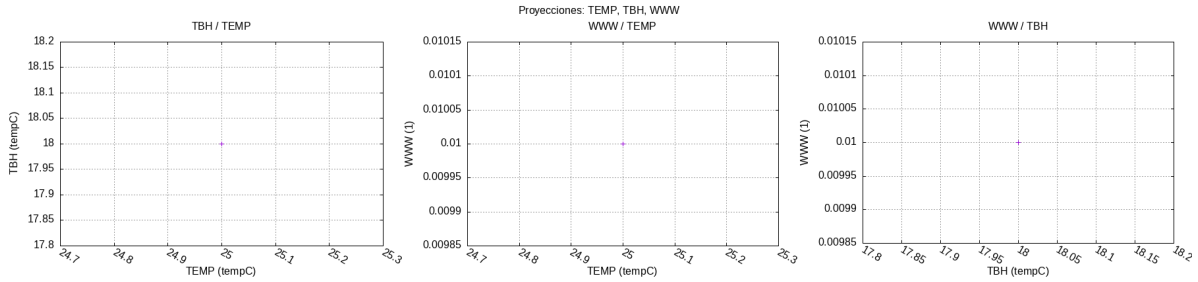


Figura 4.7: Gnuplot: un estado aislado en tres proyecciones, sin tiempo añadido.

Con tres variables se dibujan tres proyecciones bidimensionales, más un panel. Cada proyección contiene el mismo estado en coordenadas diferentes. No se agrega T ni se trazan curvas psicrométricas que no estén en los datos. Con una sola fila conviene usar puntos; una serie de líneas no contiene un segmento si no hay dos extremos. El recorrido de proyecciones convierte ese caso de una sola fila en un marcador visible; con las descripciones históricas se debe pedir puntos explícitamente.

Opción de pares	Uso
`--columnasA,B,C`	Nombres exactos o índices desde 1; entre 2 y 12 columnas, sin duplicados.
`--fuenteID`	Selecciona la fuente cuando una descripción .ds/DAT estructurada tiene varias.
`--bloqueN`	Selecciona un bloque, contando desde 1.
`--formatospng,eps,pdf,svg`	Formatos deseados; en pares el predeterminado es solamente PNG.
`--directorioDIR`	Lugar de salida; no abre ventanas.

Para n columnas hay $n(n-1)/2$ proyecciones. El programa reparte los gráficos en paneles limitados, no intenta encajar indefinidamente todas las combinaciones en una única imagen. Un .ds histórico no se convierte implícitamente para este modo: usar su tabla o producir el modelo estructurado. El corpus `corpus/psicro-punto/` conserva además el caso psicrométrico que motivó esta distinción entre un estado y una simulación temporal.

4.10. CSV, fechas, horas y zona horaria

```
T,A
2026-10-09T12:00:00Z,20
2026-10-09T12:30:00Z,22
2026-10-09T13:00:00Z,21
```

```
{"GLOBAL":{"simplot":{
  "version":2,
  "fuentes":[{"id":"MEDIDAS","archivo":"calendario.csv"}],
  "graficos":[{"id":"CALENDARIO","titulo":"Instantes UTC",
    "ejes":{"x":{"escala":"temporal","zona":"UTC"},"y":{"escala":"lineal"}},
    "series":[{"id":"A","tipo":"puntos","fuente":"MEDIDAS","x":{"nombre":"T"},"y":{"
    nombre":"A"}}]}],
  "paneles":[],"salida":{"formatos":["png"],"ventanas":false}
}}}
```

```
simplot -graf=4 -sinventana calendario.dat
```

Tabular2 convierte los instantes a segundos epoch Unix. El eje `temporal` de Simplot formatea esos números como calendario. El ejemplo usa fechas con Z y presentación UTC; por ello es reproducible en máquinas con zonas distintas.

Entrada o ajuste	Significado
Fecha y hora sin zona	Se usa la zona de la máquina, incluyendo la configuración TZ del proceso.
Fecha y hora con Z o desplazamiento	El desplazamiento explícito identifica el instante.
Fecha sin hora	Medianoche de esa fecha, según la interpretación de zona aplicable.
Hora sin fecha	Duración desde medianoche; no contiene por sí sola una fecha epoch.
Eje temporal sin `zona`	Presentación en hora local de la máquina.
`"zona":"UTC"`	Presentación UTC, como en el ejemplo.
Tiempo relativo desde el arranque	Eje lineal en segundos; no calendario salvo que se agregue un origen real fuera de Simplot.

El formato numérico del eje temporal es segundos epoch, no días ni minutos. Si los metadatos de origen declaran minutos epoch, la conversión a segundos precede al dibujo. No pedir minutos como unidad final de un eje calendario. En datos separados por espacios, usar T entre fecha y hora para mantener un solo campo; en CSV puede usarse un campo entre comillas. Para horas ambiguas por cambio estacional, escribir el desplazamiento explícito. Las horas locales inexistentes deben rechazarse, no ajustarse sin aviso.

4.11. Comparar corridas, usar logaritmos y exportar cuatro formatos

```
# T A
1 2
10 20
```

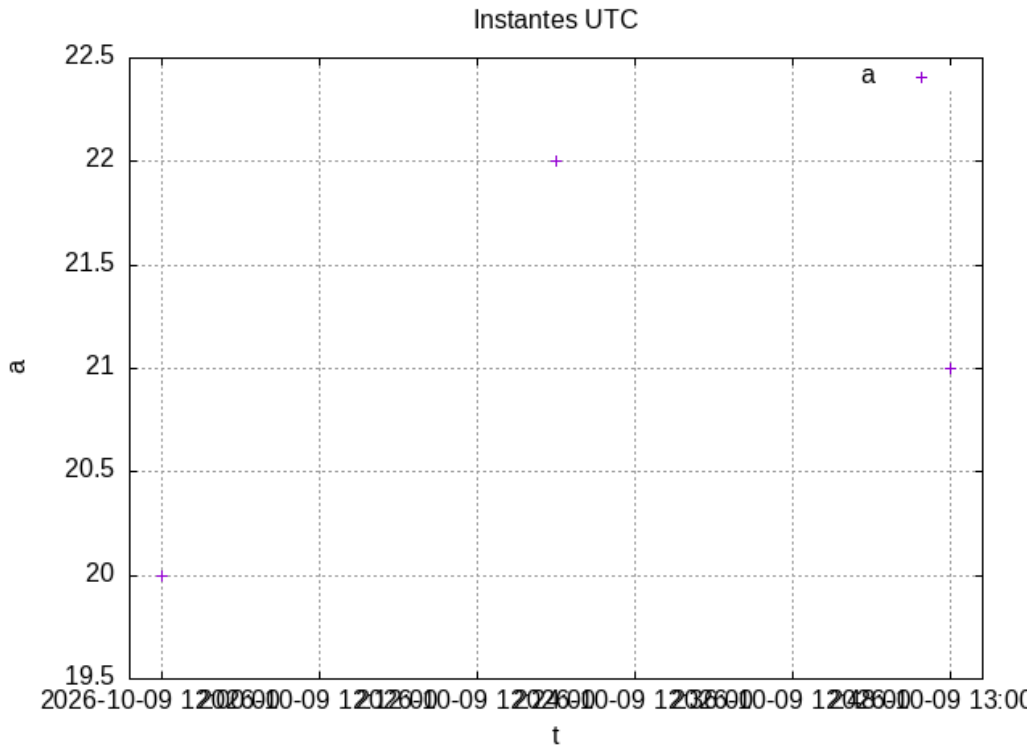


Figura 4.8: CSV convertido a epoch por Tabular2 y presentado como calendario UTC.

```
# T A
1 3
10 30
```

```
{"GLOBAL":{"simplot":{
  "version":2,
  "fuentes":[
    {"id":"UNO","archivo":"bloques.simul","bloque":{"indice":1}},
    {"id":"DOS","archivo":"bloques.simul","bloque":{"indice":2}}
  ],
  "graficos":[{"id":"CORRIDAS","titulo":"Dos corridas independientes",
    "ejes":{"x":{"escala":"log","base":10,"invertido":true},"y":{"escala":"lineal"}},
    "series":[
      {"id":"A1","tipo":"lineas","fuente":"UNO","x":{"nombre":"T"},"y":{"nombre":"A
    "},"leyenda":"Corrida 1"},
      {"id":"A2","tipo":"puntos","fuente":"DOS","x":{"nombre":"T"},"y":{"nombre":"A
    "},"leyenda":"Corrida 2"}
    ]}],
  "paneles":[],"salida":{"formatos":["png","eps","pdf","svg"],"ventanas":false}
}}}
```

```
simplot -graf=4 -sinventana bloques.dat
```

Las dos fuentes apuntan al mismo archivo, pero seleccionan bloques diferentes. El resultado superpone las corridas, sin concatenarlas ni interpolarlas a una malla común. También podrían apuntar a dos archivos distintos con muestreos diferentes. Cada serie toma X e Y de su propia fuente; no se permite combinar X de un archivo con Y de otro dentro de una sola serie sin

preparar previamente una tabla coherente.

El eje X del ejemplo es logarítmico en base 10 y está invertido. Invertir un eje no invierte el orden de las filas. Un rango debe escribirse con mínimo menor que máximo; la inversión se indica con el booleano correspondiente. Todos los valores elegidos para un eje logarítmico deben ser positivos, incluso los que luego queden fuera del encuadre. Limitar el rango no sanea datos inválidos.

Se producen PNG, EPS, PDF y SVG. En el modelo estructurado la selección de bloques por índice funciona; la selección por nombre está prevista en el modelo, pero el renderizador aún la rechaza. Si hay varios bloques, seleccionarlos expresamente. No confundir este mecanismo con `arch_var` del motor histórico, que describe familias de archivos.

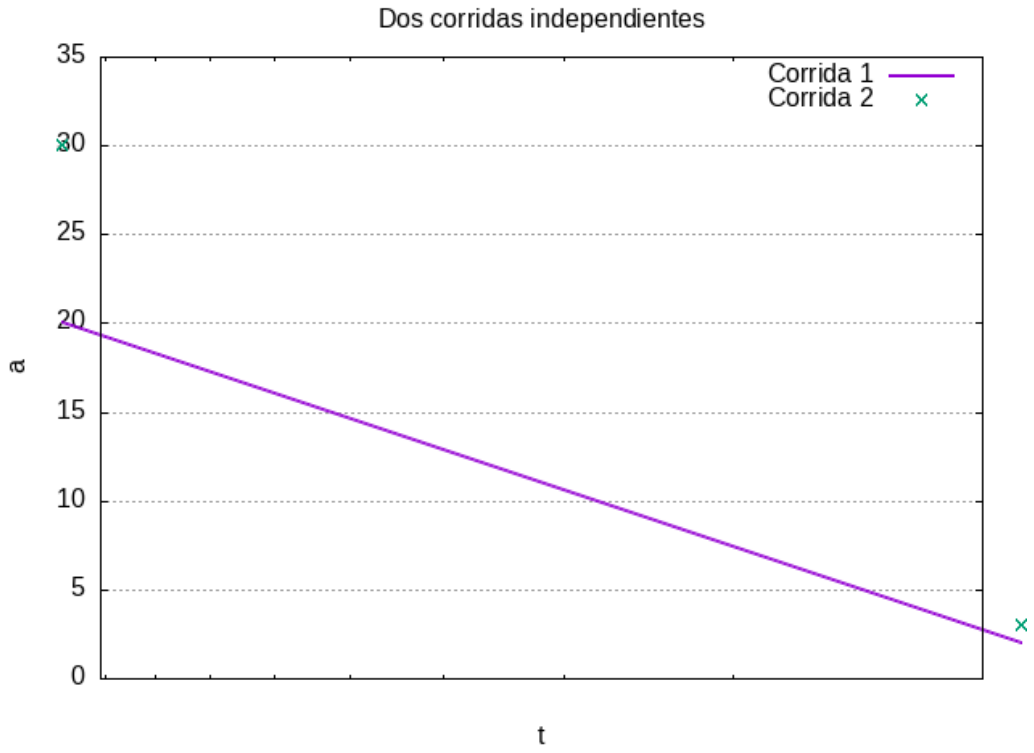


Figura 4.9: Dos bloques independientes; abscisa logarítmica e invertida.

4.12. Conservar descripciones históricas

Orden o recurso histórico	Función y precaución
archivo, variables, magnitudes, unidades	Fuentes y metadatos de la descripción histórica. No asumir que todas tienen traducción al modelo estructurado.
rangox, rangoy	Encuadre; la forma histórica usa dos puntos, mientras la sintaxis de los ejemplos nuevos usa coma. No es un filtro numérico.
mostrarjuntos	Componer gráficos en un panel.
selec	Seleccionar filas en un intervalo cerrado de una columna. Conserva el orden; se validan dependencias y ciclos.

Orden o recurso histórico	Función y precaución
arch_var, variaciones	Familias de archivos y rótulos de corridas. Las tablas existentes no exigen SCEPTRE.
funcion	Referencia analítica para Gnuplot; no resuelve el modelo ni integra ecuaciones.
guardar, nomostrar	Recursos históricos para conservar guiones o controlar presentación. No son órdenes de borrado.

El corpus conserva ejemplos de cada familia, con pruebas específicas en `tests/lenguajes.t`, `tests/seleccion.t` y las demás pruebas asociadas. No toda variante documentada hace años tiene equivalencia en el recorrido común o cobertura individual. Ejecutar primero el ejemplo histórico conocido, conservar su salida y comparar al migrarlo. No ensayar entradas históricas no confiables como si el lenguaje fuera una zona aislada de seguridad.

Capítulo 5

Referencia del modelo y de la interfaz Perl

5.1. El DAT como frontera entre programas

Un documento JSON contiene la descripción gráfica en `GLOBAL.simplot`. Sus campos son un contrato, no instrucciones Perl ni un fragmento libre de Gnuplot. Un productor puede construir el DAT directamente sin generar antes un .ds. El ejemplo `ejemplos/bloques.dat` es una descripción completa que no necesita archivo .ds para dibujarse.

El contrato estructurado se identifica internamente por `version:2`, un número JSON, no una cadena. Los campos desconocidos se rechazan. `null` no sustituye a un campo opcional ausente y los booleanos deben ser `true` o `false`, no 0, 1 ni textos. Se exige que los números sean finitos. La validación devuelve una copia del modelo y no modifica al llamador.

Objeto	Campos principales y condiciones
Modelo	Obligatorios: version, fuentes, graficos, paneles. Opcionales: salida, resultados, pares.
Fuente	id, archivo; opcional bloque. La ruta se resuelve respecto del documento, no de un directorio global implícito.
Gráfico	id, ejes, series; opcional titulo. Dos ejes, x e y, y al menos una serie.
Serie	id, tipo, fuente, x, y; opcional leyenda. El tipo determina si se unen muestras o se dibujan marcadores.
Selector de columna	`{"indice":1}` o `{"nombre":"T"}`; exactamente uno. Índices desde 1; nombres exactos, sin ambigüedad.
Panel	id, graficos; opcionales titulo y columnas. Referencia gráficos, no otros paneles; sin repeticiones.
Salida	formatos obligatorio cuando existe el objeto; ventanas opcional. Formatos png, eps, pdf, svg, sin duplicados; ventanas es booleano.

Los identificadores comienzan por letra ASCII o guion bajo y continúan con letras, dígitos, guion bajo, punto o guion. Son sensibles a mayúsculas. Las fuentes tienen su espacio de nombres; los gráficos y paneles comparten otro. Los identificadores de serie son únicos dentro del modelo, no solamente dentro de cada gráfico. La leyenda vacía permite suprimir ese texto.

Campo del eje	Valores y significado
escala	lineal, log o temporal; obligatorio.
etiqueta	Texto del eje; independiente del nombre interno de la columna.
unidad	Unidad de presentación. En temporal solo s; se necesita un origen convertible.
rango	Dos números finitos en orden creciente. En log, ambos positivos.
invertido	Booleano JSON; invierte la orientación del eje, no la tabla.
base	Solo log; positiva y distinta de 1. Si falta, se usa 10.
zona	Solo temporal; local o UTC. Si falta, se usa la zona local.

Los campos `resultados` y `pares` describen expansiones de gráficos a partir de columnas. No son expresiones numéricas arbitrarias. Para construir esos objetos desde otro programa consultar las estructuras completas y ejemplos en `doc/DAT-SIMPLOT-V2.md`, `lib/SimPlot/Modelo.pm`, `Resultados.pm` y `Pares.pm`; para una selección ordinaria es más simple generar RESULTADOS en .ds o usar la CLI de pares. Los datos de tabla no se incorporan automáticamente al DAT: al archivar un trabajo deben guardarse también sus fuentes.

5.2. Qué se expresa en .ds y qué requiere DAT

Medio	Prestaciones actuales
.ds (modelo estructurado)	RESULTADOS, UNIDADES y PARAPLOT con fuentes, series, ejes y combinación de gráficos.
DAT estructurado	Además: selección de bloque, eje logarítmico/temporal/invertido, número de columnas de panel y formatos de salida.
CLI de pares	Proyecciones automáticas de columnas seleccionadas, fuente y bloque explícitos cuando corresponde.
Compatibilidad histórica	Funciones, selección, familias de archivos y diálogo interactivo incorporados al recorrido estructurado de documentos; el corpus comprueba tanques, colector, recortes y superposiciones.

Un DAT que contenga el modelo gráfico puede dibujarse sin volver al .ds. Un DAT del sistema que carezca de `GLOBAL.simplot` puede necesitar el lector con la opción gráfica para completar el modelo desde OUTPUTS; no se debe afirmar que todo DAT arbitrario ya es una descripción gráfica. El documento original no se sobrescribe durante esa adaptación.

5.3. API por contexto: uso en lotes y servicios

```
use strict;
use warnings;
use SimPlot::Contexto;

# Ejecutar desde una copia de este directorio.
my $contexto = SimPlot::Contexto->new(base=>'.', directorio=>'salida');
```

```
for my $archivo (qw(comparacion.M.dsa calendario.dat bloques.dat)) {
    my $r = eval { $contexto->ejecutar($archivo) };
    die "No se pudo dibujar $archivo: $@" if $@;
    die "$archivo: $r->{errores}" if $r->{estado};
    print "$archivo -> $r->{directorio}\n";
}
```

```
perl -I/ruta/simplot/lib lote.pl
```

El contexto fija las rutas del trabajo. No modifica el directorio actual, los argumentos ni el entorno del programa llamador. Cada dibujo obtiene un directorio exclusivo. Es posible procesar varios trabajos consecutivos o crear contextos independientes para clientes diferentes; no reutilizar para ello las antiguas variables globales de los módulos históricos.

Método	Contrato de uso
`new(base=>...,directorio=>...)`	base debe existir; resuelve rutas del contexto. Se pueden configurar lector, interpola, gnuplot y simplot.
`leer(archivo)`	Lee un documento y devuelve su estructura. Los errores son excepciones.
`dibujar(modelo,documento,ventanas=>0)`	Dibuja el modelo estructurado tomando documento como referencia de las rutas; devuelve directorio, guion y archivos.
`ejecutar(archivo)`	Elige el recorrido estructurado cuando corresponde; en otro caso ejecuta el motor histórico aislado.
`ejecutar_historico(archivo,opciones=>[...])`	Ejecuta otro proceso Perl con un directorio propio; opciones admitidas limitadas a las previstas para ejecución por lotes.

`ejecutar` devuelve motor, directorio, archivos, estado, codigo, senal, salida y errores; el recorrido estructurado agrega el guion de dibujo. `estado` es el estado POSIX del proceso y `codigo` el código de salida decodificado. Una validación puede lanzar una excepción antes de producir ese objeto. Por eso el ejemplo hace dos comprobaciones: captura la excepción con `eval` y luego revisa el estado devuelto. No basta con que exista una referencia Perl para dar por exitoso el trabajo.

El lector puede configurarse como programa o lista de argumentos, evitando construir una línea de shell. La personalización de Gnuplot desde Contexto está implementada para el modelo estructurado, no para el motor histórico. Los programas se ejecutan mediante listas de argumentos y procesos hijos; esto evita alterar los descriptores y el entorno del llamador. No implica que todos los archivos de entrada sean seguros frente a contenido malicioso.

5.4. Alcance de la reentrancia

- Modelo estructurado: validación sobre copia independiente, sin lectura de tablas ni ejecución de programas.
- Renderizador del modelo estructurado: estado por trabajo, salidas exclusivas; puede componerse desde contextos independientes.
- Implementación histórica: conserva estado global interno. La garantía de separación proviene del nuevo proceso, no de haber vuelto puras todas sus funciones.
- Las pruebas incluyen repetición, alternancia, anidamiento y procesos simultáneos. No prueban compartir objetos mutables entre hilos Perl ni modificar globalmente TZ o PATH durante el dibujo.

- No se matan otros Gnuplot del usuario. `-kill` se rechaza; `-nokill` se conserva como compatibilidad sin esa acción destructiva.

Los resultados de una ejecución real no desaparecen al destruir el contexto. El cliente decide cuándo ya los consumió y puede borrar su directorio exacto. En caso de error pueden quedar salidas parciales útiles para diagnóstico; no interpretar la presencia de un PNG como prueba suficiente de éxito de todo un lote. Tampoco se deben borrar directorios amplios para limpiar una corrida.

Capítulo 6

Gnuplot: motor gráfico, archivos y personalización

6.1. Qué aporta y qué no expone Simplot

Gnuplot es un programa gráfico con lenguaje propio, ejecución interactiva o por guiones y múltiples terminales de salida. Su repertorio incluye muchos más estilos y representaciones que los expuestos actualmente por Simplot. El término *terminal* designa aquí un dispositivo o formato de salida, no necesariamente una ventana de consola. La disponibilidad real depende de cómo se construyó Gnuplot.[1]

Trabajo	Quién lo hace
Interpretar RESULTADOS y unidades del proyecto	Simplot, slex y Tabular2.
Trazar líneas, puntos, ejes y leyendas	Gnuplot siguiendo el guion generado.
Crear paneles	Simplot decide la composición; Gnuplot ejecuta multiplot.
Convertir temperatura o exportar la tabla	Tabular2 antes del trazado; no un cambio cosmético de etiqueta.
Decidir si una trayectoria es físicamente correcta	El productor y el usuario; una imagen plausible no valida una simulación.

6.2. Archivos de trabajo

Cada render crea un directorio `simplot-v2-XXXXXX`. Allí quedan las tablas exportadas con los identificadores de fuente, las conversiones cuando hacen falta, `graficos.gnu` y los archivos de imagen. El subdirectorio `convertidas/` contiene las tablas preparadas para las unidades de presentación. La exportación usa 17 cifras significativas para conservar la precisión de los números disponibles: no recupera cifras ausentes del archivo original.

Los PNG son imágenes rasterizadas; EPS, PDF y SVG permiten los flujos vectoriales correspondientes al terminal. El renderizador usa terminales png, postscript eps, pdfcairo y svg. Si un terminal falta, se debe corregir la dependencia o elegir otro formato: no hay sustitución silenciosa de PDF por PNG. Los nombres de imagen derivan de los identificadores de gráficos y paneles.

Se dibujan también los gráficos individuales que integran un panel.

6.3. Volver a ejecutar o adaptar un guion

```
gnuplot --version
gnuplot -e 'set terminal'
gnuplot /ruta/simplot-v2-XXXXXX/graficos.gnu
```

La última orden vuelve a escribir los archivos de salida indicados por el guion. Para personalizarlo, conservar el original, copiarlo y cambiar también los destinos de imagen antes de ejecutarlo. Las rutas generadas son absolutas: trasladar el directorio de trabajo sin corregir el guion puede dejarlo apuntando a la ubicación anterior. Para archivar un caso portátil es preferible guardar entradas y descripción y regenerar; para una instantánea exacta, guardar además guion, tablas exportadas, imágenes y versiones de las herramientas.

```
set terminal png
set output '/ruta/salida/ejemplo.png'
set xlabel 'Tiempo (s)'
set ylabel 'Temperatura (tempC)'
plot '/ruta/salida/MEDIDAS.simul' using 1:2 with lines title 'Interior'
```

El fragmento es ilustrativo: el guion real contiene también reinicios de estado, formatos, rangos, codificación y composición. Las opciones `using1:2`, `withlines` y `withpoints` corresponden a selección de columnas y estilo de representación. Agregar un ajuste estadístico, filtros o fórmulas a mano produce una personalización externa; no convierte esas funciones en parte del contrato estructurado ni de sus pruebas.

6.4. Ventanas, ejecución sin pantalla y seguridad

En lotes usar `-graf=4-sinventana`; en Contexto se ejecuta sin ventanas. Los formatos de archivo no requieren una sesión gráfica interactiva. La ruta que abre ventanas usa X11 cuando se solicita: no debe suponerse que siempre estará disponible en un servidor o contenedor. Las opciones históricas de diálogo no tienen traducción automática al modelo estructurado.

El modelo estructurado no acepta un campo de órdenes Gnuplot libres. Se escapan rutas y textos y se rechazan caracteres de control y comillas invertidas. Aun así, Simplot no es una sandbox: lee rutas, ejecuta herramientas y puede convivir con configuraciones del usuario de Gnuplot. El motor histórico tiene además lenguajes de expresiones más permisivos. Un servicio público necesita aislamiento del sistema operativo, límites de recursos y restricciones de archivos, no solamente validación del JSON.

6.5. Diagnóstico por capas

Síntoma	Comprobar antes de cambiar código
No encuentra una tabla	Resolver su ruta respecto del .ds/DAT y comprobar que se copió junto con la descripción.
Columna inexistente o ambigua	Nombres exactos, mayúsculas, cabecera exportada e índices desde 1.

Síntoma	Comprobar antes de cambiar código
Error de conversión	Unidad de origen real, unidad destino y uso correcto de tempC/tempF frente a intervalos.
Curva vacía o punto invisible	Número de filas, estilo de puntos, límites del eje y valores positivos si es logarítmico.
Fecha inesperada	Distinguir epoch de duración; revisar zona del dato, TZ y zona de presentación.
Se mezclan corridas	Declarar bloque por índice o separar fuentes, sin concatenar silenciosamente.
Falla PDF/SVG	Terminales disponibles de Gnuplot y diagnóstico del proceso; probar un formato disponible.
Código de salida no cero	Revisar el error completo, modelo, tabla exportada y guion; no confiar en una imagen parcial.
Cambió una entrada histórica al migrar	Comparar el recorrido histórico con el común y revisar las prestaciones pendientes de migrar.

Para aislar la lectura de datos puede ejecutarse `interpola--exportar-simularchivo`. Si se redirige la salida, usar otro nombre: nunca el mismo archivo de entrada. Conservar el diagnóstico, la entrada mínima, las versiones y el guion al informar un fallo. La corrección debe incorporar un caso al corpus para que el problema no reaparezca.

Capítulo 7

Limitaciones actuales y extensiones posibles

7.1. Matriz de alcance

Tema	Límite actual y alternativa
Tipos de gráfico del modelo estructurado	Líneas y puntos bidimensionales, con barras de error opcionales en X/Y a partir de límites dados. Pendientes barras de categorías, bandas rellenas, mapas, superficies y gráficos 3D.
Migración pendiente	El traductor al modelo estructurado todavía no cubre todas las órdenes históricas. Es trabajo pendiente, no una limitación deseada: el objetivo es un único PARAPLOT que conserve esas prestaciones e incorpore todas las capacidades del modelo estructurado.
Configuración .ds	Ejes especiales, bloques y formatos aún requieren DAT. No inventar órdenes .ds que el lector no reconoce.
Bloques	Selección por índice operativa; nombre previsto, pero render no lo implementa.
Tiempo	Epoch y zonas local/UTC. Marcas de calendario sencillas, no un planificador completo de intervalos por meses/años o cambios estacionales.
Unidades	Conversión con origen conocido; no inferencia física de unidades faltantes. Evitar mezclar unidades en un eje sin destino explícito.
Categorías y texto	No hay contrato general de celdas textuales como categorías. El soporte ampliado de texto y de huecos corresponde primero a Tabular2.
Datos faltantes	No se promete interpolación automática ni una política universal de NaN, huecos o valores ausentes.
Grandes tablas	Lectura y preparación actuales en memoria; procesos externos por operación. No hay garantías de streaming, reducción visual ni rendimiento para tamaños arbitrarios.

Tema	Límite actual y alternativa
Estilos y maquetación	Personalización limitada: sin esquema tipado completo de colores, espesores, símbolos, fuentes, ejes secundarios o anotaciones.
Interactividad	Diálogo histórico de selección; no editor visual moderno, panel web ni selección enlazada entre gráficos.
Precisión	Se preserva la disponible al exportar; no se reconstruye precisión perdida ni se valida por dibujar la exactitud de un solver.
Seguridad	No es sandbox ni servicio multiusuario listo para exponer entradas no confiables.

7.2. Procesamiento numérico: ampliarlo en Tabular2

La dirección acordada es mantener una frontera: Simplot pide o consume resultados numéricos; Tabular2 realiza las operaciones generales. La presencia de filtros en el recorrido histórico no justifica copiar esos algoritmos al motor nuevo. La política está registrada en `../tabular2/doc/PLAN-PROCESAMIENTO-NUMERICO.md` y `doc/PLAN-FUNCIONAL.md`.

Familia	Próximo trabajo y condiciones
Interpolación e inversa	Revisar y exponer lo existente mediante una interfaz reutilizable; especificar dominio, extrapolación, monotonicidad y ambigüedad de la inversa.
Media y RMS	Definir si son sobre muestras o ponderadas en la coordenada; no confundir muestras irregulares con intervalos iguales.
Integral	Precisar coordenada, límites, unidades del resultado y tratamiento de huecos.
Remuestreo	Hacer explícita la malla destino, el método y los límites; no remuestrear al superponer corridas sin que se pida.
Funciones adicionales	Diferenciación, suavizado, filtros, FFT, ajustes y estadísticas: evaluar y priorizar allí, con pruebas numéricas propias.
Texto y celdas ausentes	Pendiente: conservar texto y definir cómo usar valores numéricos anteriores y posteriores cuando existan. Fijar primero qué ocurre en bordes y huecos consecutivos.

Esta tabla es una agenda, no una afirmación de que todas las funciones ya se puedan solicitar desde Simplot. El primer paso es auditar las operaciones existentes de Tabular2 y exponer interpolación, inversa, media, RMS, integral y remuestreo con un contrato compartido. Toda operación debería devolver valores y metadatos, además de un error distinguible cuando no puede calcularse; no basta con producir una columna de aspecto plausible.

7.3. Extensiones del graficador, sin cambiar su responsabilidad

El lenguaje mejorado debe ser más fácil de entender, más potente y expresivo, y compatible con los PARAPLOT antiguos: no debe obligar a reescribir documentos válidos. Los casos habituales

tendrán formas breves; los complejos, opciones explícitas y componibles. Las formas antiguas y nuevas se traducirán al mismo modelo, conservando el significado histórico. Cada simplificación se probará con ejemplos equivalentes, sin sustituir los originales del corpus.

La evolución será de un solo PARAPLOT, no de dos lenguajes alternativos. Las versiones identifican contratos de datos y compatibilidad durante la transición. Primero se completará la migración del lenguaje histórico; después se expondrán en .ds todas las capacidades que ya admite el DAT estructurado. Se incorporarán además ideas gráficas de Observable Plot: capas combinables, propiedades visuales asociadas a columnas, leyendas y paneles por categorías, sin incorporar sus transformaciones numéricas al graficador. El orden, los requisitos y las pruebas de cierre están en `doc/PLAN-FUNCIONAL.md`, sección «PARAPLOT único».

1. Completar el puente .ds–DAT para las capacidades ya operativas: bloques, escalas, formatos y composición. Prueba de aceptación: la entrada .ds y su DAT equivalente generan las mismas series y unidades.
2. Añadir estilos tipados: colores, tamaño de punto, grosor, trazos y tipografía, con valores validados. No introducir una cadena arbitraria de código Gnuplot como atajo.
3. Las barras de error X/Y con límites absolutos ya están implementadas. Quedan bandas rellenas y, si se decide admitirlos, deltas explícitos. La estimación de incertidumbres sigue fuera de Simplot.
4. Diseñar ejes secundarios con asociación explícita de cada serie. Nunca colocar magnitudes distintas sobre un eje común solamente porque los números entran en el mismo rango.
5. Mejorar calendario y paneles: elección adaptativa de marcas, rótulos no superpuestos, dimensiones y composición reproducibles.
6. Incorporar barras, categorías, mapas o superficies únicamente cuando existan contratos de datos apropiados en Tabular2. Una nube de puntos no define por sí sola una malla ni una interpolación superficial.
7. Evaluar rendimientos con corpus grande antes de agregar cachés o lectura incremental; medir memoria, tiempo y fidelidad visual de cualquier reducción de datos.

Para cada ampliación cambiar conjuntamente modelo, productor slex cuando corresponda, renderizador, ejemplos, pruebas y manual. Conservar el contrato anterior o introducir una versión nueva si la interpretación cambia. Las pruebas deben verificar datos exportados y comandos relevantes, además de que exista una imagen.

7.4. Trabajo de infraestructura y seguridad

- Registrar versiones de herramientas y procedencia de fuentes en una descripción de ejecución facilitaría repetir figuras años después.
- Ofrecer un modo de archivo portátil evitaría las rutas absolutas del guion actual; debería copiar fuentes autorizadas y no seguir rutas arbitrarias sin control.
- Un servicio necesita plazos máximos, límites de filas y memoria, limpieza por trabajo y aislamiento; no compartir credenciales o el entorno completo de un usuario entre clientes.
- Una interfaz numérica más eficiente con Tabular2 podría reducir procesos y lecturas repetidas, pero debe conservar una API estable y separación de errores. No reintroducir SWIG o dependencias circulares sin una necesidad medida.
- La publicación coordinada debe garantizar que las dependencias mínimas sean descargables antes de promover un paquete como estable.

Capítulo 8

Comparación con otras herramientas

8.1. Elegir por el trabajo, no por una lista de nombres

Simplot ocupa un lugar específico: automatizar gráficos de tablas científicas del ecosistema Simusol conservando sus metadatos y descripciones. Una herramienta general puede ofrecer más estilos y seguir necesitando un adaptador para .ds, RESULTADOS y unidades del proyecto. A la inversa, la conveniencia de ese adaptador no convierte a Simplot en sustituto universal de un entorno científico o editor de figuras.

Herramienta	Diferencia útil para decidir
Gnuplot	Motor programable interactivo y por lotes, con más estilos y terminales que los expuestos aquí. Usarlo directamente si se necesita toda su expresividad; usar Simplot para describir fuentes y unidades del proyecto sin escribir a mano cada guion.[1]
Matplotlib	Biblioteca Python organizada en figuras, ejes y objetos gráficos, con interfaces programables y amplio control de presentación. Es apropiada dentro de un análisis Python; Simplot evita exigir ese entorno para el flujo Perl/.ds/DAT existente. No se compararon rendimientos.[2]
Observable Plot	Biblioteca JavaScript declarativa basada en marcas, escalas y transformaciones; admite facetas y composición orientada a visualización web. Aporta ideas para separar representación y datos. Simplot actual no ofrece una interfaz web equivalente.[3]
Veusz	Aplicación científica con edición gráfica y uso como módulo Python, gráficos 2D/3D e importación de varios formatos. Conviene cuando importa editar interactivamente la figura; Simplot prioriza descripciones y lotes reproducibles, sin editor visual propio.[4]
Plot2	Aplicación científica para OS X con edición de datos, macros, scripts Perl y prestaciones como errores, ajustes y procesamiento numérico. Su documentación orienta posibles mejoras; no se probó su ejecución ni se propone convertirla en dependencia del proyecto.[5]

No se afirma que ninguna de estas herramientas lea de fábrica los contratos .ds/DAT específicos

de Simplot ni que tenga idéntica política de unidades. Tampoco se midieron calidad de imagen, consumo de memoria o velocidad comparables. Antes de reutilizar código de otro proyecto hay que revisar su licencia y compatibilidad: una idea funcional documentada no autoriza copiar una implementación.

8.2. Nivel de abstracción respecto de PARAPLOT

La comparación tiene dos ejes independientes: **nivel de abstracción**, cuánto detalle debe expresar el usuario, y **etapa de la cadena**, qué tarea realiza la herramienta. Una herramienta puede cubrir varias etapas. El medio o formato de salida no determina ninguna de estas dos coordenadas.

1. Generar datos.
2. Procesar datos.
3. Definir qué dibujar.
4. Hacer el dibujo.
5. Representar el dibujo.

Herramienta	Etapa de la cadena	Nivel de abstracción
Siru, SCEPTRE y Calcula	1	Modelos, variables y ecuaciones del sistema; productores de datos, no lenguajes gráficos.
Tabular2 u otros	2	Operaciones sobre tablas, no pedidos de dibujo.
PARAPLOT	3	Alto nivel, especializado en fuentes, variables, series y agrupamientos.
Simplot con Gnuplot	4	Realiza el pedido de PARAPLOT: Simplot lo organiza y Gnuplot hace el dibujo.
D3 en el recorrido de Puyuspa	5	Operaciones gráficas bajo la interfaz de Puyuspa; no describe el pedido con un lenguaje como PARAPLOT.
Visor o impresora	5	Interpreta el dibujo recibido para presentarlo; no redefine qué variables graficar.
Observable Plot	2 (opcional), 3 y 4	Declarativo en la parte gráfica, semejante a PARAPLOT.
Matplotlib	3 y 4	API de figuras, ejes y objetos; generalmente más detallada que PARAPLOT.
Veusz	2 (opcional), 3 y 4	Documento y propiedades de componentes gráficos.
Plot2	2 (opcional), 3 y 4	Interfaz de gráficos y macros; nivel de las macros pendiente de evaluar.

```
Etapa 1 -> datos -> etapa 2 (opcional) -> datos preparados
                                                    |
                                          + etapa 3: pedido de dibujo
```

```
                    |
preparacion del pedido
                    |
etapa 4: hacer el dibujo
                    |
etapa 5: representar el dibujo
```

La descripción es una entrada adicional, no un cálculo sobre la tabla. Siru, SCEPTRE y Calcula ocupan el mismo lugar como productores de datos, aunque Siru utilice Calcula internamente. En la etapa 4 se considera Simplot con Gnuplot como una unidad: Simplot organiza el pedido y utiliza Gnuplot para hacer el dibujo, no son alternativas entre sí dentro de este recorrido. La etapa 5 presenta el dibujo: allí ubicamos D3 dentro de la interfaz de Puyuspa, un visor o una impresora. Hacer el dibujo y representarlo son tareas distintas, aunque una aplicación pueda reunirlas. Por ejemplo, un SVG puede ser el resultado de la etapa 4 y la entrada de la etapa 5; el formato no sustituye a la etapa.

Estos lugares describen los papeles en los recorridos considerados, no restricciones universales de cada herramienta: D3 también puede participar en la construcción del dibujo y Gnuplot puede abrir una salida interactiva. Observable Plot abarca varias etapas; eso no lo coloca por encima de PARAPLOT como lenguaje gráfico. No se presupone una cadena Gnuplot–D3 en Simplot.

Para comparar cómo dibujan se parte de los mismos datos preparados. Las transformaciones de Observable Plot quedan fuera: aquí corresponden a Tabular2 u otras herramientas. Simplot recibe valores y límites de error; no estima incertidumbres ni calcula estadísticas para dibujarlos.

El nivel de abstracción indica qué debe especificar el usuario: relaciones entre datos y gráficos, objetos de una figura o detalles de su construcción. No mide calidad ni cantidad de funciones. PARAPLOT es un lenguaje específico de alto nivel: declara fuentes, variables, series y agrupamientos sin programar cada trazo. El graficador prepara los ejes y el dibujo a partir de ese pedido.

Herramienta y nivel	Relación con PARAPLOT
Simplot/PARAPLOT: descripción del problema gráfico	Referencia: fuentes, variables, series y organización de gráficos, integradas con los metadatos y unidades del proyecto.
Observable Plot: nivel semejante a PARAPLOT	Declara qué dibujar mediante datos, marcas, canales x/y, escalas y facetas. Su API JavaScript evita construir elementos SVG uno por uno. Es más general en vocabulario visual, no por ello de mayor abstracción ni compatible con PARAPLOT.[3]
Gnuplot: instrucciones para hacer el dibujo	Describe columnas o expresiones, estilos, ejes, paneles y salida. Simplot traduce los pedidos de PARAPLOT a esas instrucciones. En el recorrido ocupa una etapa posterior a PARAPLOT, no un puesto inferior.[1]
Matplotlib: nivel generalmente inferior a PARAPLOT	Para reproducir el mismo pedido exige programar figuras, ejes y llamadas de trazado en Python. Sus funciones de alto nivel simplifican gráficos particulares y su API permite controlar objetos individuales; no es un lenguaje específico de pedidos como PARAPLOT.[2]

Herramienta y nivel	Relación con PARAPLOT
Veusz: nivel comparable en el documento gráfico	Describe datos, curvas, ejes y paneles mediante componentes y propiedades. Se acerca a PARAPLOT al configurar una figura, aunque la estructura del documento es más explícita. La interfaz visual y la programación de componentes no tienen el mismo grado de detalle.[4]
Plot2: nivel comparable en la interfaz de usuario	Permite elegir datos y configurar gráficos sin programar cada trazo. Sus macros y scripts automatizan tareas, pero su equivalencia con PARAPLOT no fue verificada: no se asigna a ese lenguaje un nivel mayor o menor sin un análisis específico.[5]
D3: nivel inferior a PARAPLOT	Combina escalas, ejes, generadores de formas y elementos web mediante JavaScript. Es también más elemental que las órdenes de gráficos de Gnuplot. Ofrece control fino, pero la capa que expresa un pedido completo debe construirla la aplicación.[6]

Esta clasificación es una interpretación de las interfaces documentadas. Observable Plot es el referente más directo como interfaz declarativa de alto nivel; Gnuplot expresa las instrucciones con las que se realiza el pedido de PARAPLOT. Veusz y Plot2 ofrecen una abstracción comparable al trabajar con documentos gráficos. Matplotlib tiene operaciones de trazado de alto nivel, pero necesita una capa de aplicación para reproducir ese contrato. D3 puede servir para implementar tal capa, no la aporta por sí mismo. Coincidir en nivel no implica tener las mismas capacidades, unidades o formatos.

8.2.1. De PARAPLOT al dibujo: una etapa posterior

PARAPLOT expresa el pedido en términos del problema: qué variables representar, de qué fuentes y cómo agruparlas. Simplot interpreta ese pedido y prepara las instrucciones concretas para Gnuplot. Al usar Gnuplot directamente, el usuario asume más decisiones sobre columnas o expresiones, estilos, ejes, disposición y salida. No significa que deba escribirlas todas: Gnuplot también tiene valores predeterminados.

La relación en la cadena es temporal y funcional: primero se define qué dibujar con PARAPLOT (etapa 3), después Simplot con Gnuplot hace el dibujo (etapa 4). No se dice que Gnuplot esté por debajo de PARAPLOT. Comparar cuánto detalle exige cada lenguaje es otra pregunta, independiente de su posición en el recorrido.

En esta comparación, PARAPLOT describe la intención, Gnuplot controla el gráfico y D3 permite construirlo a partir de piezas web más elementales. Es una comparación conceptual, no una cadena de ejecución: Simplot usa Gnuplot, no Gnuplot seguido de D3; Puyuspa utiliza D3 por otro camino.

8.2.2. Cómo leer la comparación de las otras herramientas

- Observable Plot: semejante a PARAPLOT en intención declarativa. La diferencia principal es el dominio: gramática visual general frente a pedidos integrados con los datos de Simusol.
- Matplotlib: generalmente por debajo para este trabajo, porque el usuario programa la figura y sus ejes. Una llamada breve no convierte toda su API en un lenguaje de pedidos.
- Veusz: comparable al describir el documento final; más detallado al manipular su jerarquía de componentes. No se fuerza una posición única para ambas interfaces.

- Plot2: comparable en las operaciones que ve el usuario; queda pendiente evaluar el nivel de su lenguaje de macros.
- D3: por debajo al construir gráficos, pero una aplicación como Puyuspa puede ocultar esos detalles y ofrecer una interfaz de alto nivel.

La referencia es siempre el mismo trabajo: expresar una selección de datos y su presentación. Ninguna clasificación establece que PARAPLOT sea superior para todos los gráficos o que tenga más prestaciones.

8.2.3. Qué significa la gramática de Observable Plot

Observable Plot es una biblioteca, no un lenguaje textual con un parser equivalente al de PARAPLOT. Su gramática visual se expresa con llamadas y objetos JavaScript. Una figura combina marcas; cada marca vincula datos con propiedades visuales mediante canales.[3, 7]

Pieza	Papel
Datos	Tabla que recibe cada marca.
Marcas	Líneas, puntos, barras u otras representaciones.
Canales	Asignación de campos a x, y, color o tamaño.
Escalas	Conversión de valores de datos a posiciones y atributos visuales.
Capas	Varias marcas en la misma figura.
Facetas	Subgráficos repetidos según categorías de los datos; no equivalen a cualquier composición de paneles.
Transformaciones	Operaciones como agrupamiento o agregación antes del dibujo.

Un equivalente visual básico del pedido anterior, suponiendo que `Plot` esté importado y `medidas` sea una tabla de objetos con campos T, A y B, es:

```
const grafico = Plot.plot({
  marks: [
    Plot.line(medidas, {x: "T", y: "A", stroke: "red"}),
    Plot.line(medidas, {x: "T", y: "B", stroke: "blue"})
  ]
});
document.body.append(grafico);
```

No hay bucle que dibuje cada segmento ni construcción manual de ejes. El ejemplo compara la selección de series; no implementa la lectura de archivos, las conversiones de unidades ni el resto del contrato de PARAPLOT. Observable Plot puede usarse fuera de la plataforma Observable.

8.3. D3 en Puyuspa

Puyuspa 6 y 7 ya utilizan D3 v7 para representar gráficos SVG en el navegador. En `calcula/puyuspa-6.0/` y `calcula/puyuspa-7.0/`, `psicro.html` y `sistema.html` cargan la copia local `d3.v7.min.js`. En `sistema.html` se emplean selecciones, escalas lineales/logarítmicas, ejes y generadores de líneas para construir las vistas de los datos calculados.

El nivel de aplicación lo aporta Puyuspa: el usuario elige el sistema y sus variables sin escribir D3. Esa interfaz específica no equivale a que D3 interprete PARAPLOT. Simplot usa Gnuplot; Puyuspa tiene este camino de dibujo en el navegador, además del recorrido de simulación con

Siru. Siru llama a Simplot por defecto al terminar y le transmite descripción gráfica y datos por pipe; la opción `-nosimplot` anula esa llamada. Incorporar D3 a la comparación no lo agrega como dependencia de Simplot ni anuncia un cambio de motor.

8.4. Programas del propio árbol: no confundir los nombres

Nombre	Papel y evaluación conservada
Plot histórico	Módulos Perl antiguos archivados en `viejos/plot/`. Se revisaron para recuperar comportamiento útil, no se adoptaron como una dependencia externa activa del motor nuevo.
SD2 de SCEPTRE	Conversor/posprocesador vinculado a resultados de ese simulador. Su revisión aporta casos de bloques, corridas, nombres y precisión; no obliga a instalar SCEPTRE para usar Simplot.
Observable Plot	Proyecto JavaScript externo; no es el directorio Plot histórico.
Plot2	Aplicación de Michael Wesemann; tampoco es el paquete Perl archivado.

La evidencia detallada, módulos examinados y pruebas históricas están en `doc/REVISION-GRAFICADORES.md`. La matriz de funciones y prioridades está en `doc/PLAN-FUNCIONAL.md`. Las extracciones de manuales de Simusol conservan solamente los fragmentos relevantes para gráficos, resultados y unidades en `doc/manual-extraido/`; no se presentan como documentación actual de todas las variantes.

8.5. Qué adoptar de esas comparaciones

- De los modelos declarativos: separar fuentes, transformaciones, marcas y escalas, con validación antes del dibujo.
- De los editores científicos: errores, bandas, anotaciones y control de publicación, incorporados como campos estables y reproducibles.
- De los entornos numéricos: errores explícitos y pruebas de las operaciones, pero implementadas en Tabular2 cuando sean generales.
- De los conversores de simulación: conservar bloques, precisión y procedencia sin imponer al graficador el simulador de origen.
- Del lenguaje de Gnuplot: reutilizar sus terminales y trazado en lugar de implementar otro motor de dibujo. Distinguir siempre lo que el backend sabe hacer de lo que el contrato de Simplot permite pedir.

Capítulo 9

Mapa del código, corpus y mantenimiento

Archivo o directorio	Responsabilidad
`scripts/simplot.pl`	Entrada de consola y selección del camino de ejecución.
`lib/SimPlot/Dato.pm`	Lectura del documento y relación con slex.
`lib/SimPlot/Modelo.pm`	Tipos, referencias y validación estructural.
`lib/SimPlot/Resultados.pm`	Expansión de resultados, metadatos y reglas de presentación.
`lib/SimPlot/Pares.pm`	Proyecciones bidimensionales y paneles automáticos.
`lib/SimPlot/Render.pm`	Tabular2, comprobación de datos, guion Gnuplot y salidas.
`lib/SimPlot/Contexto.pm`	API por trabajo y aislamiento de la ejecución histórica.
`lib/SimPlot/Proceso.pm`	Ejecución de procesos sin alterar al llamador.
`lib/Simusol/`	Soporte activo del camino histórico; no confundirlo con las copias archivadas.
`corpus/`	Casos autónomos, procedencia y ejemplos recuperados.
`tests/`	Regresiones del modelo, motores, unidades, instalación y distribución.
`doc/manual/ejemplos/`	Entradas pequeñas que coinciden con los listados de este manual.
`viejos/`	Material histórico para consulta, no una ruta que deba agregarse a la instalación activa.

9.1. Cómo mantener documentación y ejemplos coherentes

1. Antes de prometer una prestación, identificar su camino real: compatibilidad histórica, modelo común, lector slex, Tabular2 o Gnuplot.
2. Añadir una entrada mínima al corpus o a los ejemplos del manual. Mantener los originales históricos con su procedencia cuando sean la evidencia de un fallo.
3. Comprobar estados de salida, valores y unidades, no solamente la existencia de un archivo gráfico.
4. Ejecutar `makecheck`; para cambios de distribución, también `makedistcheck`. Los saltos de

pruebas deben quedar explicados.

5. Regenerar el PDF con `make-Cdoc/manual`. Revisar figuras, advertencias de LaTeX y enlaces, además de que la compilación termine.
6. Actualizar el changelog y los índices. Conservar los resultados históricos identificados como tales; no reemplazarlos por cifras nuevas sin contexto.
7. Para paquetes nuevos, incrementar la versión y repetir la validación de instalación. Actualizar documentación local no equivale a publicar paquetes ni a modificar la web.

9.2. Documentación complementaria

Documento	Para profundizar
`doc/GUIA-USO.md`	Resumen rápido de las formas de uso.
`doc/DAT-PARAPLOT.md`	Contrato y transporte de descripciones históricas.
`doc/DAT-SIMPLOT-V2.md`	Referencia detallada del modelo estructurado.
`doc/REENTRANCIA.md`	Decisiones de aislamiento y pruebas de contextos.
`doc/PRUEBAS.md`	Verificaciones, problemas encontrados y resultados fechados.
`doc/INSTALL.txt`, `doc/PAQUETES.md`	Instalación, formatos y distribución coordinada.
`doc/PLAN-FUNCIONAL.md`	Prioridades y evolución de la implementación.
`doc/README.md`	Índice general, páginas históricas y reglas de sincronización con la web.

Bibliografía

[1] Gnuplot, documentación oficial 6.0: *Introduction* y *Terminals*. https://gnuplot.sourceforge.net/docs_6.0/loc69.html; https://gnuplot.sourceforge.net/docs_6.0/Terminals.html.

[2] Matplotlib, *Quick start guide*. https://matplotlib.org/stable/users/explain/quick_start.html.

[3] Observable Plot, *Marks* y *Transforms*. https://observablehq.github.io/plot/features/marks; https://observablehq.github.io/plot/features/transforms.

[4] Veusz, presentación y documentación oficial. https://veusz.github.io/.

[5] Michael Wesemann, *Plot2*. https://apps.micw.org/apps/plot2/.

[6] D3, *What is D3?*: primitivas gráficas y relación con Observable Plot. https://d3js.org/what-is-d3.

[7] Observable Plot, *Plots* y *Facets*. https://observablehq.github.io/plot/features/plots; https://observablehq.github.io/plot/features/facets.

Fuentes externas consultadas el 9 de octubre de 2026. La comparación es documental; no se ensayaron estas aplicaciones ni se compararon sus tiempos de ejecución. Las referencias a archivos del proyecto son relativas a la raíz de Simplot, salvo indicación contraria.