

Standard Tipografici ed Architettura Operativa per la Visualizzazione Dati di Laboratorio

Manuale Tecnico e Guida alle API dei Moduli `plot_ttypo` e `chi_squared_2d_map`

Laboratorio di Fisica Sperimentale e Computazionale

21 ottobre 2026

Sommario

Questo manuale definisce le specifiche implementative, le convenzioni tipografiche e i protocolli d'uso dei moduli grafici `plot_ttypo` e `chi_squared_2d_map`. L'infrastruttura standardizza la produzione di figure scientifiche pronte per la pubblicazione su riviste internazionali (es. *Physical Review*, *European Physical Journal*). Vengono documentate in dettaglio: l'integrazione di $\text{\LaTeX}$ reale con font Palatino (`mathpazo`), la gestione scalare e tipografica degli ordini di grandezza tramite `format_sci_axis`, il controllo indipendente delle scale lineari e logaritmiche con tacche centrate `inout` e griglie calibrate in `apply_academic_ticks_and_grid`, e l'architettura a griglia asimmetrica per le mappe bidimensionali profilate di confidenza $\Delta\chi^2$.

Indice

1	Introduzione e Specifiche Tipografiche	2
1.1	Standard Tipografico di Laboratorio	2
2	Inizializzazione Globale: <code>setup_academic_style</code>	2
2.1	Gerarchia dei Parametri Dimensionali	3
3	Normalizzazione Scientifica degli Assi: <code>format_sci_axis</code>	4
3.1	Dettaglio dei Parametri	4
3.2	Regola di Chiamata Critica	4
4	Gestione di Assi, Tacche e Griglie: <code>apply_academic_ticks_and_grid</code>	4
5	Topologia dei Layout di Pubblicazione	7
5.1	Grafico a Pannelli Accostati (Dati + Residui)	7
5.2	Standard per Barre d'Errore e Legende	7
6	Mappe 2D Profilate di Confidenza: <code>chi2_map_pipeline</code>	7
6.1	Architettura del Canvas a Quattro Riquadri	7
6.2	Significato Geometrico dei Contorni e dei Livelli	8
6.3	Effetto Contorno (<i>Halo</i>) sulle Scritte	8
7	Guida Operativa: Esempi di Codice Pronti all'Uso	9
7.1	Esempio 1: Fit RLC con Residui e Notazione Scientifica	9
7.2	Esempio 2: Diagramma di Bode Semilogaritmico con Residui in dB	11
7.3	Esempio 3: Grafico a Doppia Scala Logaritmica (Log-Log)	11

1 Introduzione e Specifiche Tipografiche

La presentazione visiva di misure fisiche ed elaborazioni numeriche richiede lo stesso rigore applicato alla modellizzazione teorica e all'analisi statistica. Nelle librerie grafiche ad alto livello (come `matplotlib` nelle sue configurazioni di default) si riscontrano sistematicamente difetti strutturali:

1. **Disomogeneità dei font:** il testo degli assi usa font sans-serif di sistema, mentre le espressioni matematiche usano font TeX standard (Computer Modern), generando contrasti visivi sgradevoli.
2. **Cifre ridondanti e troncamenti casuali:** valori grandi o piccoli vengono stampati con 5 o 6 zeri continui (es. 90000, 105000) oppure le etichette alternano cifre intere e decimali misti (1, 1.05, 1.1), violando la coerenza di lettura.
3. **Tacche orientate verso l'interno o asimmetriche:** le tacche `inside` entrano nel canvas oscurando i dati periferici, mentre le tacche `outside` collidono con le etichette numeriche.
4. **Pannelli residui disgiunti:** l'inserimento di spaziature vuote tra il pannello del fit e quello dei residui disperde l'allineamento visivo orizzontale punto-per-punto.

I moduli descritti in questo manuale risolvono questi problemi alla radice attraverso routine modulari ad alta riusabilità.

1.1 Standard Tipografico di Laboratorio

La libreria adotta come riferimento il carattere **Palatino** per tutte le componenti del grafico:

- **Testo descrittivo, legende e titoli:** Palatino tondo e corsivo;
- **Formule, grandezze fisiche e simboli greci:** font matematico `mathpazo`;
- **Unità di misura:** sempre in carattere tondo (`\mathrm{...}`), separate dal valore numerico o dal moltiplicatore da uno spazio sottile (`\,`).

2 Inizializzazione Globale: `setup_academic_style`

La funzione `setup_academic_style()` interviene direttamente sul dizionario globale `matplotlib.rcParams`, forzando la compilazione TeX per ogni singola stringa testuale.

```
1 def setup_academic_style(  
2     textsize: float = 11.0,  
3     labelsize: float = 11.5,  
4     ticksize: float = 10.0,  
5     legendsize: float = 9.5,  
6     titlesize: float = 11.5,  
7 ) -> None:  
8     plt.rcParams.update({  
9         "text.usetex": True,  
10        "font.family": "serif",  
11        "font.serif": ["Palatino", "URW Palladio L", "TeX Gyre Pagella"],  
12        "text.latex.preamble": r""  
13            \usepackage{amsmath}  
14            \usepackage{amssymb}  
15            \usepackage{mathpazo}  
16        """,  
17        "font.size": textsize,  
18        "axes.labelsize": labelsize,  
19        "xtick.labelsize": ticksize,  
20        "ytick.labelsize": ticksize,
```

```
21     "legend.fontsize": legendsize,
22     "figure.titlesize": titlesize,
23 }
```

Listing 1: Inizializzazione dello stile Palatino LaTeX.

2.1 Gerarchia dei Parametri Dimensionali

I valori di default garantiscono la leggibilità su stampe in formato A4 a singola o doppia colonna:

- `labelsize = 11.5pt`: etichette delle coordinate x e y ;
- `ticksize = 10.0pt`: valori numerici sui tick primari;
- `legendsize = 9.5pt`: testo informativo delle legende;
- `textsize = 11.0pt`: annotazioni interne e box statistici.

3 Normalizzazione Scientifica degli Assi: `format_sci_axis`

Nelle misure fisiche reali, i valori spaziano spesso su molti ordini di grandezza (es. frequenze in decine di kilohertz o tensioni in millivolt). La funzione `format_sci_axis` risolve l'estrazione automatica del fattore di scala $\times 10^n$, garantendo che:

1. Le cifre numeriche visualizzate sulle tacche rimangano nell'intervallo $[0.1, 99]$;
2. Il numero di cifre decimali sia omogeneo su tutto l'asse;
3. L'unità di misura non collida con l'ambiente matematico e risieda sempre in tondo (Hz, V, s).

```
1 def format_sci_axis(  
2     ax: plt.Axes,  
3     vals: Sequence[float] | np.ndarray,  
4     base_label: str,  
5     axis: str = "y",  
6     unit: str = "",  
7     min_exp: int = 2,  
8     dec: int = 2,  
9 ) -> str:  
10     arr = np.asarray(vals, dtype=float)  
11     non_zero = arr[arr != 0]  
12     amax = np.max(np.abs(non_zero)) if len(non_zero) > 0 else 0.0  
13     esp = int(math.floor(math.log10(amax))) if amax > 0 else 0  
14  
15     u_clean = unit.strip().replace("$", "")  
16     if u_clean.startswith(r"\mathrm{") and u_clean.endswith("}"):   
17         u_clean = u_clean[len(r"\mathrm{"):-1]  
18  
19     unit_math = rf"\mathrm{{{u_clean}}}" if u_clean else ""  
20  
21     if abs(esp) >= min_exp:  
22         scale = 10.0**esp  
23         fmt = ticker.FuncFormatter(lambda v, _: f"{v / scale:.{dec}f}")  
24         getattr(ax, f"{axis}axis").set_major_formatter(fmt)  
25  
26         if unit_math:  
27             suffix = rf"\, [${\times 10^{{{esp}}}}\ {unit_math}$]"  
28         else:  
29             suffix = rf"\, [${\times 10^{{{esp}}}}$]"  
30         return f"{base_label}{suffix}"  
31  
32     suffix = rf"\, [${unit_math}$]" if unit_math else ""  
33     return f"{base_label}{suffix}"
```

Listing 2: Implementazione della funzione `format_sci_axis`.

3.1 Dettaglio dei Parametri

3.2 Regola di Chiamata Critica

Attenzione: In Matplotlib, l'applicazione di locator come `AutoMinorLocator` o `MaxNLocator` resetta il formattatore scalare di default. Di conseguenza, `format_sci_axis()` deve essere invocata **dopo** aver configurato le tacche e le griglie con `apply_academic_ticks_and_grid()`.

4 Gestione di Assi, Tacche e Griglie: `apply_academic_ticks_and_grid`

La funzione `apply_academic_ticks_and_grid` standardizza la geometria fisica del frame di visualizzazione. Essa introduce tre aspetti cardine:

Parametro	Tipo	Funzione e Comportamento
ax	plt.Axes	Asse Matplotlib su cui registrare il FuncFormatter.
vals	ArrayLike	Serie di valori da cui stimare l'ordine di grandezza asintotico.
base_label	str	Nome fisico della variabile (es. <code>r"Frequenza di scansione"</code>).
axis	str	Asse di applicazione: <code>'x'</code> oppure <code>'y'</code> .
unit	str	Unità di misura SI (es. <code>"Hz"</code> , <code>"V"</code>).
min_exp	int	Soglia minima sull'esponente assoluto per attivare la riscalatura (default: 2).
dec	int	Numero fisso di cifre decimali dopo la virgola (default: 2).

Tabella 1: Specifiche dei parametri di ingresso per `format_sci_axis`.

1. **Tacche bidirezionali inout:** le tacche attraversano la linea dell'asse simmetricamente, metà all'interno e metà all'esterno, evitando sia di coprire i dati periferici sia di invadere le cifre numeriche.
2. **Disaccoppiamento Lineare/Logaritmico:** è possibile impostare l'asse x e l'asse y indipendentemente su scala lineare o logaritmica.
3. **Griglia bicanale a contrasto calibrato:** griglia principale tratteggiata (`--`, colore grigio `#cccccc`) e griglia secondaria punteggiata fine (`:`, colore `#e5e5e5`).

```

1 def apply_academic_ticks_and_grid(
2     ax: plt.Axes,
3     scale_x: str = "linear", # "linear" oppure "log"
4     scale_y: str = "linear", # "linear" oppure "log"
5     major_len: float = 8.0,
6     minor_len: float = 4.0,
7     major_width: float = 0.8,
8     minor_width: float = 0.5,
9 ) -> None:
10     ax.set_xscale(scale_x)
11     ax.set_yscale(scale_y)
12
13     if scale_x == "log":
14         ax.xaxis.set_major_locator(ticker.LogLocator(base=10.0, numticks=15))
15         ax.xaxis.set_minor_locator(ticker.LogLocator(base=10.0, subs=np.arange(2,
16     10) * 0.1, numticks=15))
17         ax.xaxis.set_major_formatter(ticker.LogFormatterMathtext())
18     else:
19         ax.xaxis.set_minor_locator(ticker.AutoMinorLocator())
20
21     if scale_y == "log":
22         ax.yaxis.set_major_locator(ticker.LogLocator(base=10.0, numticks=15))
23         ax.yaxis.set_minor_locator(ticker.LogLocator(base=10.0, subs=np.arange(2,
24     10) * 0.1, numticks=15))
25         ax.yaxis.set_major_formatter(ticker.LogFormatterMathtext())
26     else:
27         ax.yaxis.set_minor_locator(ticker.AutoMinorLocator())
28
29     ax.tick_params(
30         axis="both", which="major", direction="inout",
31         length=major_len, width=major_width, colors="black",
32         top=True, right=True, bottom=True, left=True,
33     )
34     ax.tick_params(
35         axis="both", which="minor", direction="inout",
36         length=minor_len, width=minor_width, colors="black",
37         top=True, right=True, bottom=True, left=True,
38     )

```

```
37 ax.grid(True, which="major", ls="--", lw=0.6, color="#cccccc", alpha=0.85)
38 ax.grid(True, which="minor", ls=":", lw=0.4, color="#e5e5e5", alpha=0.7)
39
```

Listing 3: Implementazione di `apply_academic_ticks_and_grid`.

5 Topologia dei Layout di Pubblicazione

5.1 Grafico a Pannelli Accostati (Dati + Residui)

Nei report sperimentali, il grafico di regressione deve includere inferiormente il diagramma dei residui verticali o normalizzati (r_i o z_i).

Per garantire che l'ascissa sia rigorosamente allineata e priva di fessure, si usa `gridspec.GridSpec` forzando lo spazio verticale a zero (`hspace=0.0`):

```
1 fig = plt.figure(figsize=(6.0, 5.2))
2 gs = gridspec.GridSpec(2, 1, height_ratios=[3.2, 1.0])
3 gs.update(hspace=0.0)
4
5 ax_main = fig.add_subplot(gs[0, 0])
6 ax_res = fig.add_subplot(gs[1, 0], sharex=ax_main)
7
8 # Nasconde le etichette dell'asse X sul pannello superiore
9 ax_main.tick_params(axis="x", labelbottom=False)
10
11 # Rimuove il tick 0.00 inferiore per evitare collisioni con la linea di giunzione
12 ax_main.yaxis.set_major_locator(ticker.MaxNLocator(nbins=5, prune="lower"))
13
14 # Impostazione simmetrica dei residui con zero centrato
15 max_dev = np.max(np.abs(residui) + err_y)
16 ax_res.set_ylim(-1.35 * max_dev, 1.35 * max_dev)
17 ax_res.yaxis.set_major_locator(ticker.MaxNLocator(nbins=3, symmetric=True, prune="both"))
```

Listing 4: Costruzione del layout a due pannelli accostati.

5.2 Standard per Barre d'Errore e Legende

- **Barre d'errore pulite:** utilizzare `capsize=0` o `capsize=1.6`. L'assenza o il contenimento del trattino terminale riduce l'affollamento visivo in campioni ad alta densità di punti.
- **Legenda a cornice opaca:**

```
1 ax.legend(
2     loc="upper right", frameon=True, fancybox=False,
3     facecolor="white", edgecolor="black", framealpha=1.0,
4     borderpad=0.4, handlelength=1.4
5 )
6
```

Gli angoli retti (`fancybox=False`) e l'opacità piena (`framealpha=1.0`) impediscono alle curve sottostanti di disturbare il testo.

6 Mappe 2D Profilate di Confidenza: `chi2_map_pipeline`

Il modulo `chi_squared_2d_map.py` integra direttamente l'oggetto `RisultatiFit` prodotto da `FitPipeline`. L'obiettivo è mappare le superfici di livello del $\Delta\chi^2$ sul piano coordinato (θ_x, θ_y) e affiancare sui bordi i profili marginali 1D calcolati tramite l'algoritmo MINOS.

6.1 Architettura del Canvas a Quattro Riquadri

Per ospitare contemporaneamente la mappa bidimensionale, i due profili 1D, lo spazio informativo e la colorbar senza sbilanciare le proporzioni, la griglia viene suddivisa con `gridspec.GridSpec`:

Subplot Top (ax_t) Profilo 1D parametro x (Altezza: 1.0, Larghezza: 3.5)	Box Statistico $\text{Cov}(\theta_x, \theta_y), \rho$ (Altezza: 1.0, Larghezza: 1.0)	Vuoto	Vuoto
Mappa Centrale (ax1) $p(x, y)$ continuo + Contorni (Altezza: 3.5, Larghezza: 3.5)	Subplot Destro (ax_r) Profilo 1D parametro y (Altezza: 3.5, Larghezza: 1.0)	Pad 0.1	Colorbar (cax) Spessore: 0.15

Figura 1: Topologia dei subplot generati da `chi2_map_pipeline` con `colorbar_loc="right"`.

6.2 Significato Geometrico dei Contorni e dei Livelli

La mappa traccia simultaneamente le regioni di confidenza 2D e i vincoli 1D:

- **Contorni 2D congiunti** (linee bianche continue, tratteggiate e punteggiate):

$$\Delta\chi^2 = 2.30 \ (1\sigma_{2D}), \quad \Delta\chi^2 = 6.18 \ (2\sigma_{2D}), \quad \Delta\chi^2 = 11.83 \ (3\sigma_{2D}) \quad (1)$$

Queste curve delimitano la probabilità congiunta che entrambi i parametri cadano simultaneamente nella regione indicata (68.27%, 95.45% e 99.73%).

- **Contorno 1D marginale** (linea nera tratto-punto):

$$\Delta\chi^2 = 1.00 \ (1\sigma_{1D}) \quad (2)$$

Le rette tangenti a questa curva ellittica coincidono esattamente con le linee verticali e orizzontali rosse proiettate dagli errori MINOS singoli dei profili marginali, confermando visivamente la consistenza dell'ottimizzazione.

6.3 Effetto Contorno (*Halo*) sulle Scritte

Sulle mappe a variazione continua di intensità cromatica (es. `viridis_r`), il testo scuro rischia di perdersi sulle regioni scure, mentre il testo bianco diventa illeggibile sulle regioni chiare. Il problema viene risolto applicando un contorno tipografico dinamico (*halo*):

```

1 halo = [patheffects.withStroke(linewidth=2.5, foreground="white")]
2 clbl_2d = ax1.clabel(cs_2d, inline=True, fmt=fmt_clabel, colors="black")
3 plt.setp(clbl_2d, path_effects=halo)

```

Listing 5: Applicazione del path effect per il contrasto delle etichette di livello.

7 Guida Operativa: Esempi di Codice Pronti all'Uso

7.1 Esempio 1: Fit RLC con Residui e Notazione Scientifica

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import matplotlib.gridspec as gridspec
4 import matplotlib.ticker as ticker
5 from plot_typo import setup_academic_style, format_sci_axis,
   apply_academic_ticks_and_grid
6
7 setup_academic_style()
8
9 # Modello teorico
10 f0, A0, gamma = 105.7e3, 5.2e4, 4.5e3
11 def risonanza(f):
12     return A0 / np.sqrt(1.0 + ((f**2 - f0**2) / (gamma * f))**2)
13
14 f_dense = np.linspace(90e3, 122e3, 500)
15 y_dense = risonanza(f_dense)
16
17 rng = np.random.default_rng(42)
18 f_data = np.linspace(92e3, 120e3, 25)
19 err_y = np.full_like(f_data, 0.04 * A0)
20 y_data = risonanza(f_data) + rng.normal(0, err_y)
21 residui = y_data - risonanza(f_data)
22
23 # Layout a due pannelli
24 fig = plt.figure(figsize=(6.0, 5.2))
25 gs = gridspec.GridSpec(2, 1, height_ratios=[3.2, 1.0])
26 gs.update(hspace=0.0)
27
28 ax_main = fig.add_subplot(gs[0, 0])
29 ax_res = fig.add_subplot(gs[1, 0], sharex=ax_main)
30
31 # Pannello principale
32 ax_main.errorbar(f_data, y_data, yerr=err_y, fmt="o", markersize=3.0,
33                 color="black", ecolor="black", elinewidth=0.8, capsize=0,
34                 label=r"Dati misurati", zorder=3)
35 ax_main.plot(f_dense, y_dense, color="#1f77b4", lw=1.4, label=r"Fit teorico",
36             zorder=2)
37 ax_main.axvline(f0, color="red", ls=":", lw=1.1, label=r"$f_0 = 105.7\,\mathrm{kHz}$")
38
39 # Pannello residui
40 ax_res.errorbar(f_data, residui, yerr=err_y, fmt="o", markersize=3.0,
41                color="black", ecolor="black", elinewidth=0.8, capsize=0, zorder
42                =3)
43 ax_res.axhline(0.0, color="red", ls="--", lw=1.1)
44 ax_res.axvline(f0, color="red", ls=":", lw=1.1)
45
46 # Geometria e tacche
47 apply_academic_ticks_and_grid(ax_main, scale_x="linear", scale_y="linear")
48 apply_academic_ticks_and_grid(ax_res, scale_x="linear", scale_y="linear")
49 ax_main.tick_params(axis="x", labelbottom=False)
50
51 # Bilanciamento residui e soppressione zero critico
52 ax_res.set_yticks([-5e3, 0.0, 5e3])
53 ax_res.set_ylim(-7.5e3, 7.5e3)
54 ax_main.yaxis.set_major_locator(ticker.MaxNLocator(nbins=5, prune="lower"))
55
56 # Formattazione assi (chiamata per ULTIMA)
```

```

55 ax_main.set_ylabel(format_sci_axis(ax_main, y_dense, r"Ampiezza", axis="y", unit="
    u.a."))
56 ax_res.set_ylabel(format_sci_axis(ax_res, residui, r"Residui", axis="y", unit="u.a
    .", dec=1))
57 ax_res.set_xlabel(format_sci_axis(ax_res, f_data, r"Frequenza di scansione", axis=
    "x", unit="Hz"))
58
59 ax_main.legend(loc="upper right", frameon=True, fancybox=False,
60               facecolor="white", edgecolor="black", framealpha=1.0)
61 fig.tight_layout(pad=0.25)
62 fig.savefig("esempio_fit_residui.pdf", dpi=300, bbox_inches="tight", pad_inches
    =0.08)

```

Listing 6: Script completo per il plottaggio di un picco di risonanza con residui.

7.2 Esempio 2: Diagramma di Bode Semilogaritmico con Residui in dB

```
1 f0, Q = 10.0e3, 3.0
2 def modulo_db(f):
3     s = 1j * (f / f0)
4     H = 1.0 / (1.0 + (s / Q) + s**2)
5     return 20.0 * np.log10(np.abs(H))
6
7 f_dense = np.logspace(3, 5, 500)
8 mag_dense = modulo_db(f_dense)
9
10 f_data = np.logspace(3.1, 4.9, 22)
11 err_mag = np.full_like(f_data, 0.45)
12 mag_data = modulo_db(f_data) + rng.normal(0, err_mag)
13 residui = mag_data - modulo_db(f_data)
14
15 fig = plt.figure(figsize=(6.0, 5.2))
16 gs = gridspec.GridSpec(2, 1, height_ratios=[3.2, 1.0])
17 gs.update(hspace=0.0)
18
19 ax_main = fig.add_subplot(gs[0, 0])
20 ax_res = fig.add_subplot(gs[1, 0], sharex=ax_main)
21
22 ax_main.plot(f_dense, mag_dense, color="#1f77b4", lw=1.4, label=r"Modello  $|H(f)|$ ")
23 ax_main.errorbar(f_data, mag_data, yerr=err_mag, fmt="o", markersize=3.0,
24                 color="black", ecolor="black", elinewidth=0.8, capsize=1.6, label=
25                 =r"Dati misurati")
26 ax_main.axvline(f0, color="red", ls=":", lw=1.1, label=r"$f_0 = 10\,\mathrm{kHz}$")
27
28 ax_res.errorbar(f_data, residui, yerr=err_mag, fmt="o", markersize=3.0,
29                color="black", ecolor="black", elinewidth=0.8, capsize=1.6)
30 ax_res.axhline(0.0, color="red", ls="--", lw=1.1)
31 ax_res.axvline(f0, color="red", ls=":", lw=1.1)
32
33 apply_academic_ticks_and_grid(ax_main, scale_x="log", scale_y="linear")
34 apply_academic_ticks_and_grid(ax_res, scale_x="log", scale_y="linear")
35 ax_main.tick_params(axis="x", labelbottom=False)
36
37 max_dev = np.max(np.abs(residui) + err_mag)
38 ax_res.set_ylim(-1.35 * max_dev, 1.35 * max_dev)
39 ax_res.yaxis.set_major_locator(ticker.MaxNLocator(nbins=3, symmetric=True, prune="
40 both"))
41
42 ax_main.set_ylabel(r"Ampiezza  $[\mathrm{dB}]$ ")
43 ax_res.set_xlabel(r"Frequenza  $[\mathrm{Hz}]$ ")
44 ax_res.set_ylabel(r"Residui  $[\mathrm{dB}]$ ")
45
46 ax_main.legend(loc="lower left", frameon=True, fancybox=False,
47               facecolor="white", edgecolor="black", framealpha=1.0)
48 fig.tight_layout(pad=0.25)
49 fig.savefig("esempio_bode_residui.pdf", dpi=300, bbox_inches="tight", pad_inches
50 =0.08)
```

Listing 7: Script per grafico di Bode e residui con asse X logaritmico.

7.3 Esempio 3: Grafico a Doppia Scala Logaritmica (Log-Log)

```
1 A_theo, alpha = 3.2e-4, -1.65
2 def power_law(f): return A_theo * (f**alpha)
3
```

```

4 f_dense = np.logspace(1, 4, 400)
5 S_dense = power_law(f_dense)
6
7 f_data = np.logspace(1.2, 3.8, 16)
8 err_S = 0.12 * power_law(f_data)
9 S_data = power_law(f_data) + rng.normal(0, err_S)
10
11 fig, ax = plt.subplots(figsize=(5.8, 4.4))
12
13 ax.plot(f_dense, S_dense, color="#2ca02c", lw=1.4, label=rf"Modello  $S(f) \propto$ 
14  $f^{\{\alpha: .2f\}}$ ")
15 ax.errorbar(f_data, S_data, yerr=err_S, fmt="o", markersize=3.2,
16             color="black", ec="black", elinewidth=0.8, capsize=1.8, label=r"
17 Dati spettrali")
18
19 apply_academic_ticks_and_grid(ax, scale_x="log", scale_y="log")
20
21 ax.set_xlabel(r"Frequenza  $[\mathrm{Hz}]$ ")
22 ax.set_ylabel(r"Densit\`a spettrale  $[\mathrm{V}^2, \mathrm{Hz}^{-1}]$ ")
23
24 ax.legend(loc="upper right", frameon=True, fancybox=False,
25           facecolor="white", edgecolor="black", framealpha=1.0)
26 fig.tight_layout(pad=0.3)
27 fig.savefig("esempio_log_log.pdf", dpi=300, bbox_inches="tight", pad_inches=0.08)

```

Listing 8: Script per grafico Log-Log di leggi di potenza.