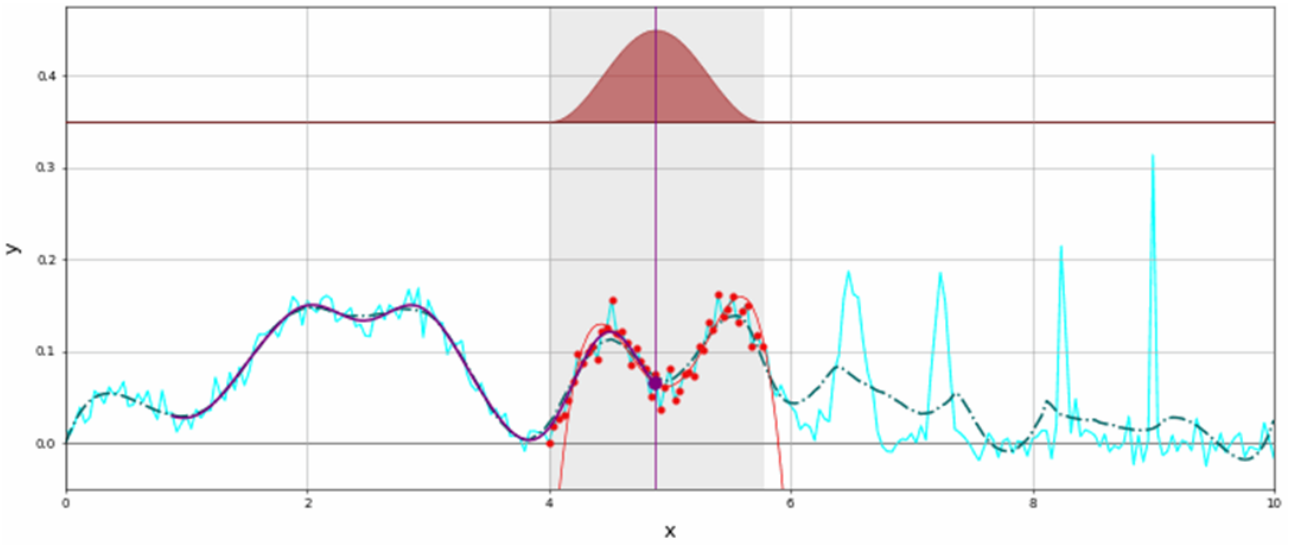
Spektral Veri Analizi

Savitzky–Golay Filtresi Uygulaması

Staj Raporu

Kadir Yıldız

github.com/lavrensiyum/MTA-Staj



İçindekiler

1 ICP-OES Spektral Verileriyle Kalibrasyon

Bu projede **ICP-OES** yöntemi ile elde edilen **Kobalt (Co)** spektral verilerinin işlenmesi için kapsamlı bir analiz gerçekleştirilmiştir.

Amaç: Çalışma kapsamında ICP-OES spektral verilerinden en yüksek kalitede ve güvenilir bilgiyi çıkarmak hedeflendi. Bu doğrultuda sinyal-gürültü oranını artırmak, pik (peak) tespit doğruluğunu yükseltmek ve kalibrasyon eğrilerinin güvenilirliğini geliştirmek amaçlandı. Ayrıca, tek parametreli sabit işleme stratejilerine kıyasla farklı spektral bölgeleri kendi karakteristiğine uygun parametrelerle ele alan *hibrit* bir yaklaşımların eğri çizmekte ne kadar performans göstediği araştırıldı.

Yapılanlar:

- Ham sinyallerde arka plan etkilerini azaltmak için *background subtraction* uygulandı.
- Sinyal örneklemesini iyileştirmek ve pürüzsüz bir spektral profil oluşturmak amacıyla *cubic spline* (PCHIP, Akima dahil) interpolasyon teknikleri ile veri artırma gerçekleştirildi.
- Gürültüyü azaltıp pik şekillerini korumak için *Savitzky–Golay* filtreleme farklı pencere uzunluğu (WL), polinom derecesi (PO) ve türev sırası (F) kombinasyonlarıyla denendi.
- 0–100 ppm aralığında farklı derişimlerde hazırlanan Co standartlarına ait verilerde, özellikle 228.615 nm dalga boyu çevresine odaklanılarak ayrıntılı pik analizi yürütüldü.
- Parametre seçiminde sistematik bir arama sağlamak üzere *grid search* ile çoklu kombinasyonlar test edilerek en iyi sonuçlar belirlendi.
- Farklı spektral alt bölgelerin farklı parametrelerle işlendiği 2-segment ve 3-segment *hibrit* işleme stratejileri geliştirildi ve tek-parça yaklaşımlarla karşılaştırıldı.

Analiz Yol Haritası:

1. **Bölüm 1. Ana Spektral Veri İşleme Pipeline Fonksiyonu**: Konsantrasyon bazlı dosya organizasyonu ve kalite kontrol adımları uygulandı.
2. **Bölüm 2. Spektral Veri Görselleştirme ve Peak Analizi Fonksiyonları**: Background subtraction, interpolasyon ve ilk filtreleme gerçekleştirildi.
3. **Bölüm 3. Arka Plan Düzeltme, İnterpolasyon ve Düzgünleştirme**: Grid search ile WL, PO, F ve interpolasyon türleri için en iyi ayarlar araştırıldı.
4. **Bölüm 4. İlk İşleme Testi ve Sonuçlarının Değerlendirilmesi**: Widget tabanlı arayüz ile parametrelerin gerçek zamanlı etkisi incelendi ve hibrit yaklaşımlar geliştirildi.
5. **Bölüm 5. Hibrit Yöntemlerin Performans Karşılaştırması**: 2-segment ve 3-segment yapılandırmalar tasarlanıp uygulandı ve tüm yöntemler Peak NRMSE, RMSE ve görsel inceleme ile değerlendirildi.

Burada kullanılacak olan kodlar, aşağıdaki GitHub reposunda mevcuttur.

GitHub Repository: https://github.com/lavrensiyum/MTA-Staj

1.1 Bölüm 1.Ana Spektral Veri İşleme Pipeline Fonksiyonu

Parça 1: Fonksiyon tanımı ve parametreler

```
def process_spectral_data(co_data: dict,
                          background_subtract: bool = True,
                          increase_points_flag: bool = True,
                          apply_smooth: bool = True,
                          factor: int = 10,
                          window_length: int = 51,
                          polyorder: int = 3) -> dict:
    """
    Tum spektral verileri isleyen ana pipeline fonksiyonu.

    Args:
        co_data: Konsantrasyon → DataFrame dictionary'si
        background_subtract: Arka plan duzeltmesi yapilsin mi?
        increase_points_flag: Interpolasyon ile veri artirma yapilsin mi?
        apply_smooth: Savitzky-Golay duzlestirme uygulansin mi?
        factor: Interpolasyon artirma faktoru
        window_length: SG filtre pencere boyutu
        polyorder: SG filtre polinom derecesi

    Returns:
        dict: Islenmis spektral veriler {konsantrasyon: DataFrame}
    """
```

Listing 1: process_spectral_data: Ana pipeline fonksiyonu tanımı

`co_data` parametresi, konsantrasyon değerlerini DataFrame'lere eşleyen bir dictionary'dir. Diğer parametreler işleme adımlarının hangilerinin uygulanacağını kontrol eder. Tüm parametreler varsayılan değerlere sahiptir, böylece fonksiyon minimum parametre ile de çağrılabilir.

Parça 2: Veri işleme döngüsü başlangıcı

```
    processed = {}

    for conc in sorted(co_data.keys()):
        df = co_data[conc]
        current_df = df.copy()
```

Listing 2: process_spectral_data: Veri işleme döngüsü

Bu parçada işleme başlangıcı yapılır. `processed` dictionary'si sonuçları saklamak için oluşturulur. Konsantrasyon değerleri sıralı olarak işlenir (`sorted` ile). Her konsantrasyon için orijinal verinin korunması için DataFrame'in bir kopyası alınır.

Parça 3: Sıralı işleme adımları

```
        # 1. Arka plan duzeltmesi (isteye bagli)
        if background_subtract and 'Background' in current_df.columns:
            current_df = perform_background_subtraction(current_df)

        # 2. Veri artirma (isteye bagli)
        if increase_points_flag:
            current_df = increase_data_points(current_df, factor=factor)

        # 3. Duzlestirme (isteye bagli)
        if apply_smooth:
            current_df = apply_smoothing(current_df, window_length=window_length, polyorder=polyorder)
```

Listing 3: process_spectral_data: Sıralı işleme adımları

- **Arka plan düzeltmesi:** Eğer `background_subtract` True ise ve DataFrame'de `Background` sütunu varsa, `perform_background_subtraction` fonksiyonu çağrılır.
- **Veri artırma:** Eğer `increase_points_flag` True ise, `increase_data_points` fonksiyonu ile veri noktaları artırılır.
- **Düzleştirme:** Eğer `apply_smooth` True ise, Savitzky-Golay filtresi ile düzleştirme uygulanır.

Bu sıralı yaklaşım, her işlemin önceki işlemin sonucu üzerinde çalışmasını sağlar.

Parça 4: Sonuç hazırlama ve döndürme

```
        # 4. Orijinal referans guvenligi
        if 'Spectrum_Original' not in current_df.columns:
            current_df['Spectrum_Original'] = np.nan

        processed[conc] = current_df

    return processed
```

Listing 4: process_spectral_data: Sonuç hazırlama ve döndürme

İşlenmiş DataFrame, konsantrasyon anahtarı ile `processed` dictionary'sine eklenir. Son olarak tüm işlenmiş veriler döndürülür.

Bu pipeline fonksiyonu sayesinde, farklı konsantrasyonlardaki spektral veriler tutarlı bir şekilde işlenebilir ve her adımın sonucu korunur.

1.2 Bolüm 2. Spektral Veri Görselleştirme ve Peak Analizi Fonksiyonlari

Bu bölümde, işlenmiş spektral verileri görselleştirmek ve peak analizi yapmak için geliştirilen üç ana fonksiyon detaylı olarak açıklandı. Bu fonksiyonlar, farklı konsantrasyonlardaki spektral verilerin karşılaştırmalı analizini ve peak pozisyonlarının karşılaştırılmasını sağlar.

Parça 1: Spektral Veri Görselleştirme Fonksiyonu - Fonksiyon Tanımı ve Parametreler

```
def plot_processed_spectra(processed_data: dict,
                           peak_range=(228.610, 228.620),
                           figsize=(18, 8)):
    """
    Islenmis spektral verileri gorsellestirir.

    Args:
        processed_data: Islenmis spektral veri dictionary'si
        peak_range: Peak bolgesi zoom araligi (nm)
        figsize: Grafik boyutu (width, height)
    """
```

Listing 5: plot_processed_spectra: Fonksiyon tanimi ve parametreler

Bu fonksiyon, işlenmiş spektral verileri iki farklı grafikte görselleştirir: tam spektrum ve peak bölgesi detay grafiği. `processed_data` parametresi, konsantrasyon anahtarları ile DataFrame değerlerini eşleyen bir dictionary'dir. `peak_range` parametresi peak bölgesinin zoom yapılacağı dalga boyu aralığını belirler. `figsize` parametresi grafik boyutunu kontrol eder.

Parça 2: Grafik Oluşturma ve Tam Spektrum Çizimi

```
    concs = sorted(processed_data.keys())
    fig, (ax_full, ax_peak) = plt.subplots(1, 2, figsize=figsize)

    # Tam spektrum grafigi
    for c in concs:
        df = processed_data[c]

        # En iyi seri secimi
        if 'Smoothed_Corrected_Spectrum' in df.columns:
            y = df['Smoothed_Corrected_Spectrum']
            series_label = "processed"
        elif 'Interpolated_Corrected_Spectrum' in df.columns:
            y = df['Interpolated_Corrected_Spectrum']
            series_label = "processed"
        else:
            y = df['Spectrum']
            series_label = "processed"

        ax_full.plot(df['Wavelength'], y, label=f"{c:.0f} ppm ({series_label})", linewidth=1.5)
```

Listing 6: plot_processed_spectra: Grafik olusturma ve tam spektrum

Bu parça, grafik yapısını oluşturur ve tam spektrum verilerini çizer. Öncelikle konsantrasyon değerleri sıralanır ve 1x2 boyutunda alt grafik yapısı oluşturulur. Her konsantrasyon için en iyi veri serisi seçilir: düzeltilmiş ve düzleştirilmiş spektrum varsa o kullanılır, yoksa ham spektrum kullanılır. Her seri farklı renk ve etiket ile tam spektrum grafiğine eklenir.

Parça 3: Ham Veri Overlay ve Peak Bölgesi Detay Grafiği

```
        # Ham veri overlay
        if hasattr(df, 'attrs') and 'Spectrum_Original' in df.attrs:
            raw_y = df.attrs['Spectrum_Original']
            raw_x = df.attrs['original_wavelength']
            ax_full.scatter(raw_x, raw_y, s=10, alpha=0.5, label=f"{c:.0f} ppm (raw)")

    ax_full.set_title('Tam Spektrum - Tum Konsantrasyonlar')
    ax_full.set_xlabel('Dalga Boyu (nm)')
    ax_full.set_ylabel('Yogunluk')
    ax_full.grid(True, alpha=0.3)
    ax_full.legend(fontsize=8)

    # Peak bolgesi detay grafigi
    lo, hi = peak_range
    for c in concs:
        df = processed_data[c]

        # En iyi seri secimi
        if 'Smoothed_Corrected_Spectrum' in df.columns:
            y = df['Smoothed_Corrected_Spectrum']
        elif 'Interpolated_Corrected_Spectrum' in df.columns:
            y = df['Interpolated_Corrected_Spectrum']
        else:
            y = df['Spectrum']

        mask = (df['Wavelength'] >= lo) & (df['Wavelength'] <= hi)
        ax_peak.plot(df['Wavelength'][mask], y[mask], label=f"{c:.0f} ppm (processed)", linewidth=2)
```

Listing 7: plot_processed_spectra: Ham veri overlay ve peak bolgesi

Bu parça, ham verilerin overlay olarak eklenmesi ve peak bölgesi detay grafiğinin oluşturulmasını sağlar. Ham veriler varsa, orijinal dalga boyu ve spektrum değerleri scatter plot olarak eklenir. Peak bölgesi için belirtilen aralıkta maskeleme yapılarak sadece o bölge çizilir. Bu sayede peak detayları daha net görülebilir.

Parca 4: Peak Tespiti Fonksiyonu - Fonksiyon Tanimi ve Ana Mantik

```
def find_peaks_in_processed_data(processed_data: dict,
                                 spectrum_col: str = 'Smoothed_Corrected_Spectrum',
                                 height=None, distance=None, prominence=None, width=None) -> dict:
    results = {}

    for conc, df in sorted(processed_data.items()):
        if spectrum_col not in df.columns:
            raise ValueError(f"Kolon bulunamadi: {spectrum_col} @ {conc} ppm")

        y = df[spectrum_col].values
        x = df['Wavelength'].values

        peaks, props = find_peaks(y, height=height, distance=distance,
                                  prominence=prominence, width=width)
```

Listing 8: find_peaks_in_processed_data: Peak tespiti fonksiyonu

Bu fonksiyon, işlenmiş spektral verilerde peak tespiti yapar. SciPy'nin `find_peaks` fonksiyonu kullanılarak farklı peak tespit kriterleri uygulanabilir. Her konsantrasyon için belirtilen spektrum serisinde peak'ler aranır ve sonuçlar dictionary yapısında saklanır.

Parça 5: Peak Tablosu Oluşturma ve Sonuç Döndürme

```
        # Peak tablosu olustur
        peak_table = pd.DataFrame({
            "Wavelength (nm)": x[peaks],
            "Peak Height": props.get('peak_heights', np.full_like(peaks, np.nan, dtype=float))
        })

        results[conc] = peak_table

    return results
```

Listing 9: find_peaks_in_processed_data: Peak tablosu olusturma

Bu parça, tespit edilen peak'ler için DataFrame tablosu oluşturur. Her peak'in dalga boyu ve yüksekliği kaydedilir. Eğer peak yüksekliği bilgisi mevcut değilse, NaN değerler kullanılır. Sonuçlar konsantrasyon anahtarları ile eşlenerek döndürülür.

Parca 6: Peak Pozisyon Karsilastirma Fonksiyonu - Fonksiyon Tanimi ve Veri Hazirlama

```
def compute_peak_distance_to_raw_max(processed_data: dict,
                                     spectrum_col: str = 'Smoothed_Corrected_Spectrum',
                                     peak_range=None) -> pd.DataFrame:
    rows = []

    for conc, df in sorted(processed_data.items()):
        # Islenmis veri serisi secimi
        if spectrum_col in df.columns:
            y_series = df[spectrum_col]
        elif 'Interpolated_Corrected_Spectrum' in df.columns:
            y_series = df['Interpolated_Corrected_Spectrum']
        else:
            y_series = df['Spectrum']
        x_series = df['Wavelength']
```

Listing 10: compute_peak_distance_to_raw_max: Peak pozisyon karsilastirma

Bu fonksiyon, islenmis ve ham veriler arasindaki peak pozisyon farklarini hesaplar. Her konsantrasyon icin en uygun veri serisi secilir ve dalga boyu ekseni belirlenir. Bu bilgiler, sonraki peak pozisyon hesaplamalari icin hazirlanir.

Parça 7: İşlenmiş Veri Peak Pozisyonu Hesaplama

```
        # Islenmis pik pozisyonu
        if peak_range is not None:
            lo, hi = peak_range
            mask_p = (x_series >= lo) & (x_series <= hi)
            if mask_p.sum() > 0:
                idx = y_series[mask_p].idxmax()
                proc_peak_wl = float(x_series.loc[idx])
                proc_peak_I = float(y_series.loc[idx])
            else:
                proc_peak_wl = np.nan
                proc_peak_I = np.nan
        else:
            idx = y_series.idxmax()
            proc_peak_wl = float(x_series.loc[idx])
            proc_peak_I = float(y_series.loc[idx])
```

Listing 11: compute_peak_distance_to_raw_max: Islenmis veri peak pozisyonu

Bu parça, işlenmiş verilerde peak pozisyonunu hesaplar. Eğer belirli bir peak aralığı belirtilmişse, sadece o aralıkta maksimum değer aranır. Aksi halde tüm spektrumda maksimum değer bulunur. Peak'in dalga boyu ve yoğunluk değerleri kaydedilir.

Parca 8: Ham Veri Peak Pozisyonu ve Fark Hesaplama

```
        # Ham veri pik pozisyonu
        raw_x = getattr(df, 'attrs', {}).get('original_wavelength', None)
        raw_y = getattr(df, 'attrs', {}).get('Spectrum_Original', None)

        if raw_x is not None and raw_y is not None:
            if peak_range is not None:
                lo, hi = peak_range
                rmask = (raw_x >= lo) & (raw_x <= hi)
                if np.any(rmask):
                    rel = int(np.argmax(raw_y[rmask]))
                    abs_idx = np.where(rmask)[0][rel]
                    raw_max_wl = float(raw_x[abs_idx])
                    raw_max_I = float(raw_y[abs_idx])
                else:
                    raw_max_wl = np.nan
                    raw_max_I = np.nan
            else:
                abs_idx = int(np.argmax(raw_y))
                raw_max_wl = float(raw_x[abs_idx])
                raw_max_I = float(raw_y[abs_idx])
        else:
            raw_max_wl = np.nan
            raw_max_I = np.nan

        # Fark hesaplamalari
        d_lambda = proc_peak_wl - raw_max_wl if (pd.notna(proc_peak_wl) and pd.notna(raw_max_wl)) else np.nan
        dI = proc_peak_I - raw_max_I if (pd.notna(proc_peak_I) and pd.notna(raw_max_I)) else np.nan
```

Listing 12: compute_peak_distance_to_raw_max: Ham veri peak pozisyonu ve farklar

Bu parça, ham verilerde peak pozisyonunu bulur ve işlenmiş verilerle karşılaştırır. Ham veriler `attrs` içinde saklanır ve aynı peak aralığı mantığı uygulanır. Dalga boyu ve yoğunluk farkları hesaplanır. Bu farklar, veri işleme kalitesini değerlendirmek için kullanılır.

Parça 9: Sonuç Tablosu Oluşturma ve Döndürme

```
        rows.append({
            'Concentration (ppm)': conc,
            'Processed Peak lambda (nm)': proc_peak_wl,
            'Processed Peak I': proc_peak_I,
            'Raw Max lambda (nm)': raw_max_wl,
            'Raw Max I': raw_max_I,
            'Delta lambda (nm)': d_lambda,
            'Delta I': dI,
            '|Delta lambda| (nm)': abs(d_lambda) if pd.notna(d_lambda) else np.nan,
            '|Delta I|': abs(dI) if pd.notna(dI) else np.nan,
        })

    return pd.DataFrame(rows)
```

Listing 13: compute_peak_distance_to_raw_max: Sonuc tablosu olusturma

Bu son parça, tüm hesaplama sonuçlarını bir DataFrame tablosunda toplar. Her konsantrasyon için işlenmiş ve ham veri peak pozisyonları, bunlar arasındaki farklar ve mutlak fark değerleri kaydedilir. Bu tablo, veri işleme performansının karşılaştırmalı analizi için kullanılır.

1.3 Bölüm 3. Arka Plan Düzeltme, İnterpolasyon ve Düzgünleştirme

Bu bölümde arka plan düzeltmesi, cubic spline ile veri noktalarının arttırılması ve Savitzky-Golay filtresi ile spektral eğrilerin düzgünleştirilmesi işlemleri detaylı olarak açıklanmaktadır.

Parça 1: Arka plan düzeltmesi

```
def perform_background_subtraction(df: pd.DataFrame) -> pd.DataFrame:
    """
    ICP-OES verilerinde arka plan duzeltmesi yapar.
    Args:
        df: Ham spektral veri
    Returns:
        pd.DataFrame: Arka plani duzeltilmis spektral veri
    """
    out = df.copy()
    out['Corrected_Spectrum'] = out['Spectrum'] - out['Background']
    if 'Spectrum_Original' not in out.columns:
        out['Spectrum_Original'] = df['Spectrum']
    return out
```

Listing 14: perform_background_subtraction: arka plan duzeltmesi

Arka plan sinyalinin ana spektrumdan çıkartılmasıyla `Corrected_Spectrum` elde edilir. Orijinal spektrum `Spectrum_Original` değişkeninde korunur.

Parça 2: Cubic spline ile veri noktalarini artirma

```
def increase_data_points(df: pd.DataFrame, factor: int = 10) -> pd.DataFrame:
    """
    Cubic spline interpolasyon ile spektral veri noktalarini artirir.
    Args:
        df: Girdi spektral veri
        factor: Artirma faktoru (varsayilan: 10x)
    Returns:
        pd.DataFrame: Interpolasyon ile artirilmis spektral veri
    """
    x = df['Wavelength'].values
    n_new = len(x) * factor
    x_new = np.linspace(x.min(), x.max(), n_new)
    res = pd.DataFrame({'Wavelength': x_new})
    cs_spec = interp.CubicSpline(x, df['Spectrum'].values)
    res['Spectrum'] = cs_spec(x_new)
    res['Interpolated_Spectrum'] = res['Spectrum']
    if 'Background' in df.columns:
        cs_bg = interp.CubicSpline(x, df['Background'].values)
        res['Interpolated_Background'] = cs_bg(x_new)
    if 'Corrected_Spectrum' in df.columns:
        cs_corr = interp.CubicSpline(x, df['Corrected_Spectrum'].values)
        res['Interpolated_Corrected_Spectrum'] = cs_corr(x_new)
    res['Spectrum_Original'] = np.nan
    res.attrs['Spectrum_Original'] = df['Spectrum_Original'].values if 'Spectrum_Original' in df.columns else
     None
    res.attrs['original_wavelength'] = x
    return res
```

Listing 15: increase_data_points: cubic spline interpolasyon

Bu adımda dalga boyu ekseni `x` daha sık örneklenir ve spektrum ile varsa arka plan/düzeltilmiş spektrum cubic spline ile `x_new` üzerine haritalanır (map func). Orijinal eksen referansları `attrs` içinde tutulur.

Parça 3: Savitzky-Golay ile düzgünleştirme

```python
def apply_smoothing(df: pd.DataFrame, window_length: int = 51, polyorder: int = 3) -> pd.DataFrame:
    """
    Savitzky-Golay filtreleme ile spektral veriyi duzlestirir.
    Args:
        df: Girdi spektral veri
        window_length: Filtre pencere boyutu (tek sayi olmali)
        polyorder: Polinom derecesi
    Returns:
        pd.DataFrame: Duzlestirilmis spektral veri
    """
    if window_length % 2 == 0:
        window_length += 1
    out = df.copy()
    if 'Interpolated_Corrected_Spectrum' in out.columns:
        out['Smoothed_Corrected_Spectrum'] = savgol_filter(
            out['Interpolated_Corrected_Spectrum'].values, window_length, polyorder
        )
    elif 'Corrected_Spectrum' in out.columns:
        out['Smoothed_Corrected_Spectrum'] = savgol_filter(
            out['Corrected_Spectrum'].values, window_length, polyorder
        )
    if 'Interpolated_Spectrum' in out.columns:
        out['Smoothed_Interpolated_Spectrum'] = savgol_filter(
            out['Interpolated_Spectrum'].values, window_length, polyorder
        )
    elif 'Spectrum' in out.columns:
        out['Smoothed_Spectrum'] = savgol_filter(
            out['Spectrum'].values, window_length, polyorder
        )
    return out
```

Listing 16: apply_smoothing: Savitzky–Golay filtreleme

Bu fonksiyon Savitzky-Golay filtresini kullanarak spektral veriyi düzgünleştirir. İşlem adımları şu şekildedir:

- **Pencere boyutu kontrolü:** Savitzky-Golay filtresi tek sayı pencere boyutu gerektirir. Eğer `window_length` çift sayı ise, bir artırılarak tek sayı yapılır.
- **Düzeltilmiş spektrum önceliği:** Fonksiyon öncelikle düzeltilmiş spektrum sütunlarını arar:
  - Eğer `Interpolated_Corrected_Spectrum` mevcutsa, bu sütun filtrelenerek `Smoothed_Corrected_Spectrum` oluşturulur
  - Aksi halde `Corrected_Spectrum` varsa, bu filtrelenerek `Smoothed_Corrected_Spectrum` oluşturulur
- **Ham spektrum işlemi:** Benzer şekilde ham spektrum için:
  - `Interpolated_Spectrum` mevcutsa, filtrelenerek `Smoothed_Interpolated_Spectrum` oluşturulur
  - Aksi halde `Spectrum` varsa, filtrelenerek `Smoothed_Spectrum` oluşturulur

Bu yaklaşım sayesinde hem düzeltilmiş hem de ham spektrum verileri aynı anda işlenebilir. İnterpolasyonlu sütunlar varsa öncelik onlara verilir çünkü daha düzgün bir veri yapısına sahiptirler.

1.4 Bölüm 4. İlk İşleme Testi ve Sonuçlarının Değerlendirilmesi

Bu bölümde, geliştirilen spektral veri işleme pipeline'ının ilk testi ve sonuçlarının detaylı değerlendirilmesi sunulmaktadır. MVP (Minimum Viable Product) parametreleri ile başlayarak, işleme sürecinin her adımı ve elde edilen sonuçlar analiz edilmektedir.

Parça 1: MVP Parametrelerinin Tanımlanması

```
# MVP (Minimum Viable Product) Parametreleri
BG_SUBTRACT = True      # Arka plan duzeltmesi yapilsin
INCREASE_POINTS = True  # Interpolasyon ile veri artirma yapilsin
SMOOTH = True           # Savitzky-Golay duzlestirme uygulansin
FACTOR = 10             # Interpolasyon faktoru (10x daha fazla nokta)
WINDOW = 51             # SG filtre pencere boyutu
POLY = 3                # SG filtre polinom derecesi
PEAK_RANGE = (228.610, 228.620)  # Peak analiz bolgesi (nm)
```

Listing 17: MVP Parametreleri: Ana işleme ayarları

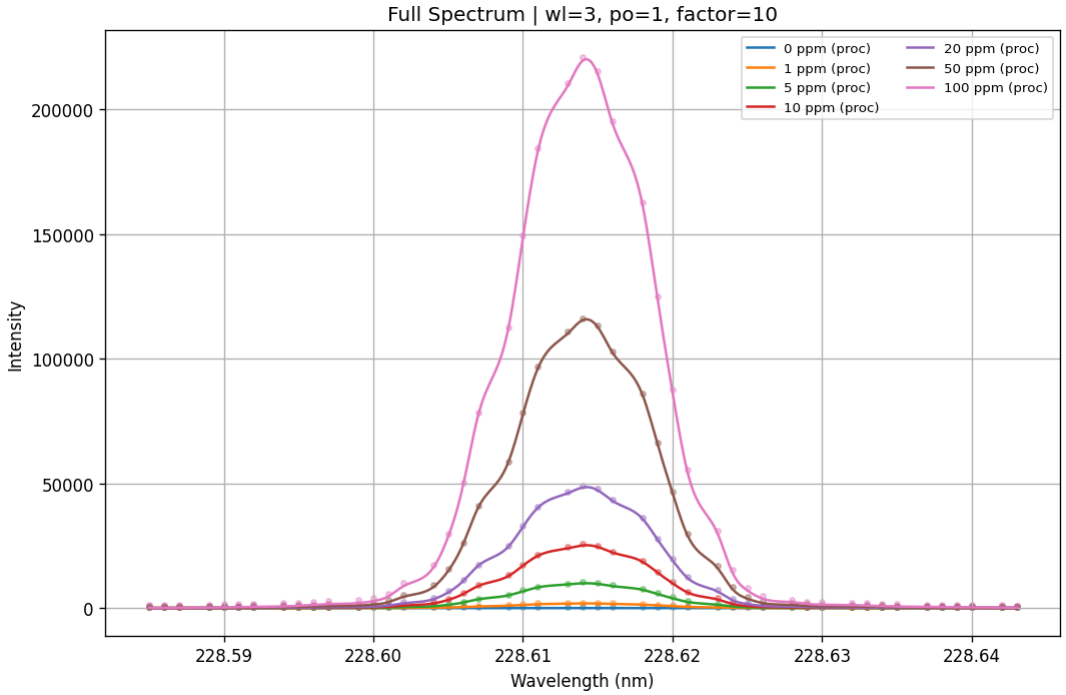
Bu MVP parametreleri, spektral veri işleme pipeline'ının temel ayarlarını tanımlar:

- **BG_SUBTRACT (True):** Arka plan düzeltmesi aktif edilmiştir. Ham spektrum verilerinden arka plan sinyali çıkartılarak gerçek analitik sinyal elde edilir.
- **INCREASE_POINTS (True):** Cubic spline interpolasyon ile veri noktalarının artırılması aktif edilmiştir.
- **SMOOTH (True):** Savitzky-Golay filtresi ile spektral verilerin düzgünleştirilmesi aktif edilmiştir. Bu işlem, gürültüyü azaltarak peak pozisyonlarının daha doğru belirlenmesini sağlar.
- **FACTOR (10):** İnterpolasyon faktörü 10 olarak ayarlanmıştır. Bu, orijinal veri noktalarının 10 katına çıkarılması anlamına gelir. Örneğin, 100 noktalı bir spektrum 1000 noktaya çıkarılır.
- **WINDOW (51):** Savitzky-Golay filtresinin pencere boyutu 51 nokta olarak belirlenmiştir. Bu değer, düzgünleştirme işleminin ne kadar geniş bir alanda yapılacağını kontrol eder. Daha büyük değerler daha fazla düzgünleştirme sağlar.
- **POLY (3):** Savitzky-Golay filtresinin polinom derecesi 3 olarak ayarlanmıştır. Kübik polinom kullanılarak hem düzgünleştirme hem de peak şeklinin korunması sağlanır.
- **PEAK_RANGE (228.610-228.620 nm):** Kobalt elementinin karakteristik emisyon çizgisinin analiz edileceği dalga boyu aralığı tanımlanmıştır. Bu dar aralık, peak pozisyonunun hassas bir şekilde belirlenmesini sağlar.

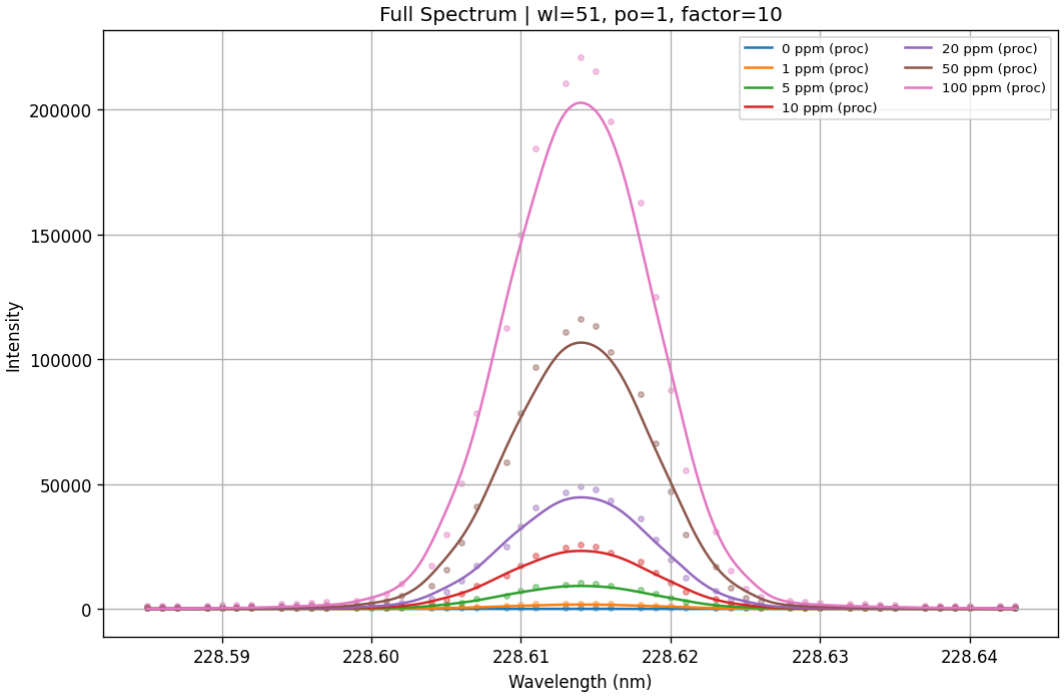
1.4.1 Windows ve Poly Değerlerinin Karşılaştırmalı Gösterimi

WL= 3 vs WL= 51

PO= 1 - PO= 1



Şekil 1: Co'nun Pik Bölgesinin Detaylı Grafiği



Şekil 2: Co'nun Pik Bölgesinin Detaylı Grafiği

1.4.2 Window Length (WL) Parametresinin Etkisi:

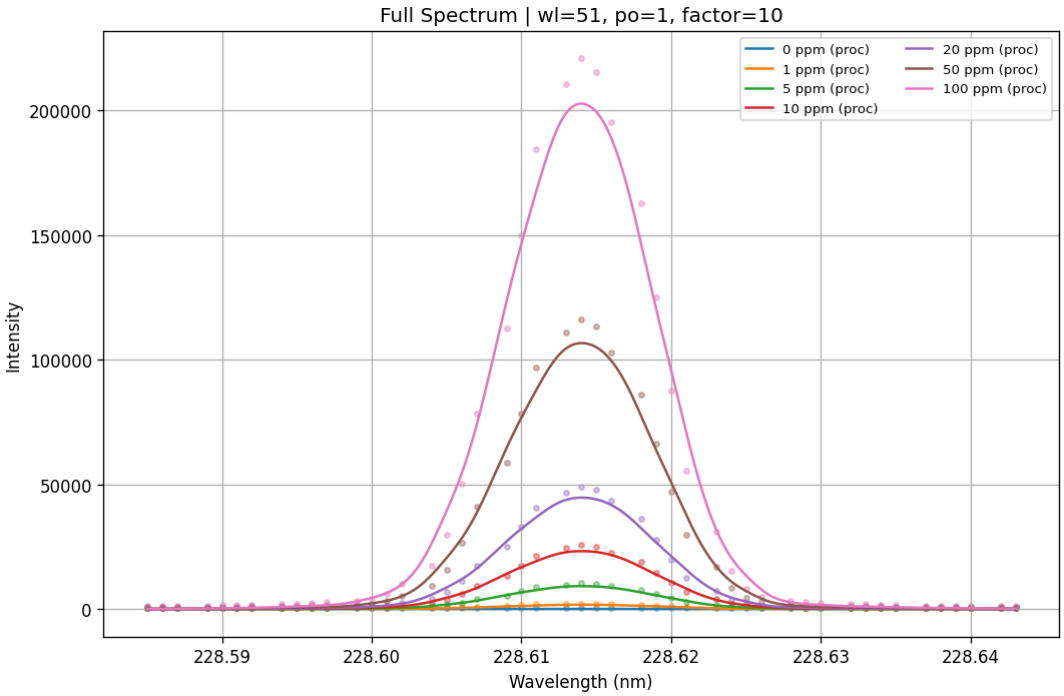
Savitzky-Golay filtresinde window length (pencere boyutu) parametresi, düzgünleştirme işleminin ne kadar geniş bir alanda yapılacağını belirler. Polinom derecesi (PO = 1) sabit tutularak WL değerinin değişiminin etkileri şu şekildedir:

- **WL = 3 (Minimum Pencere):**
  - En küçük pencere boyutu kullanılır (3 nokta)
  - Minimal düzgünleştirme etkisi sağlar
  - Orijinal verinin detayları ve gürültüsü büyük ölçüde korunur
  - Peak şekli ve pozisyonu orijinaline çok yakın kalır
  - Yüksek frekanslı gürültü büyük ölçüde filtrelenmez
- **WL = 51 (Geniş Pencere):**
  - Çok daha geniş pencere boyutu (51 nokta)
  - Güçlü düzgünleştirme etkisi yaratır
  - Gürültü önemli ölçüde azaltılır
  - Peak şekli daha düzgün ve pürüzsüz hale gelir
  - İnce detaylar kaybolabilir
  - Peak genişliği değişebilir

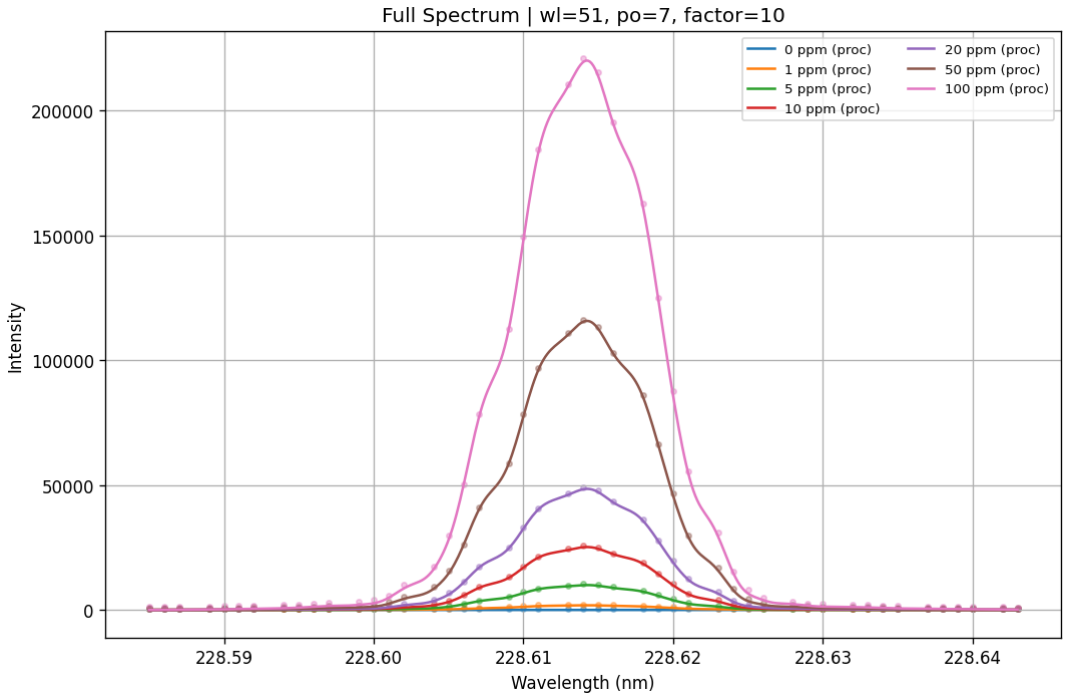
Sonuç: Spektral analiz için kullanılan WL = 51 değeri, grafikte de göründüğü üzere değeri büyüdükçe daha pürüzsüz ve daha güçlü düzgünleştirme yapmıştır. Bu her ne kadar daha tercih edilir olsa da, 1. Grafiğe göre tepe noktası kayması yaşanmış ve detaylarda büyük kayıplar yaşanmıştır.

1.4.3 PO= 1 vs PO= 5

WL= 51 - WL= 51



Şekil 3: Co'nun Pik Bölgesinin Detaylı Grafiği



Şekil 4: Co'nun Pik Bölgesinin Detaylı Grafiği

1.4.4 Polynomial Order (PO) Parametresinin Etkisi:

Savitzky-Golay filtresinde polynomial order (polinom derecesi) parametresi, düzgünleştirme işleminde kullanılan polinomun derecesini belirler. Window length (WL= 51) sabit tutularak PO değerinin değişiminin etkileri şu şekildedir:

- **PO= 1 (Lineer Polinom):**
  - Birinci derece (lineer) polinom kullanılır
  - Basit ve hızlı düzgünleştirme sağlar
  - Veri noktaları arasında lineer interpolasyon yapar
  - Keskin geçişleri yumuşatır
  - Peak şekillerini basitleştirir
  - Hesaplama açısından verimlidir
- **PO= 5 (Beşinci Derece Polinom):**
  - Beşinci derece polinom kullanılır
  - Daha karmaşık ve esnek düzgünleştirme yapar
  - Orijinal verinin eğrilik özelliklerini daha iyi korur
  - Peak şekillerini daha doğru bir şekilde temsil eder
  - İnce detayları ve eğrilikleri koruma eğilimi gösterir
  - Hesaplama açısından daha yoğun işlem gerektirir

Sonuç: Aynı spektral analiz için kullanılan PO= 5 değeri, grafiklerde de görüldüğü üzere daha yüksek derece polinom değerlerinde orijinal spektrumun eğrilik özelliklerini daha iyi korumuştur. Ancak yine de istediğimiz daha az kavisli eğriyi 1. Grafikte görebiliyoruz.

1.4.5 Çözüm:

Çözüm olarak bir Grid Araması çalıştırıldı ve tam uzay aralığında,

```
  # Alternatif: tam uzay
  WL_LIST = list(range(3, 101, 2))
  PO_LIST = [1, 2, 3, 4, 5, 6, 7]
  FACTOR_LIST = list(range(1, 61, 1))
```

Listing 18: Grid Araması İçin Uzay Parametreleri

değelerinde Peak NRMSE, RMSE, N değerleri bir `for():` döngüsüyle bir listede toplandı ve en iyi Peak NRMSE, RMSE ve N değerlerinin olduğu ilk 3 değer sıralandırıldı:

```
Arama tamamlandi. Denenen kombinasyon sayisi: 20580 | Sure: 621.7 s - 10m 21.7s

En iyi ayar:
  WL=3, PO=2, Factor=21
  Peak NRMSE=0.277% | RMSE=611.2 | N=63

Ilk 3 sonuc:
      WL    PO    Factor  Peak NRMSE    RMSE          N
80    3     2     21      0.002772      611.171362    63
620   5     4     21      0.002772      611.171362    63
1160  7     6     21      0.002772      611.171362    63
```

Listing 19: Grid Arama Sonuçları

Bu grid arama sonuçları, 20580 farklı parametre kombinasyonunun test edildiğini göstermektedir. En iyi performans WL=3, PO=2, Factor=21 parametreleriyle elde edilmiştir. Bu kombinasyon %0.277 Peak NRMSE değeri ile en düşük hata oranını sağlamıştır. İlginç olan nokta, farklı WL ve PO değerlerinin aynı Factor=21 değeriyle benzer performans göstermesidir.

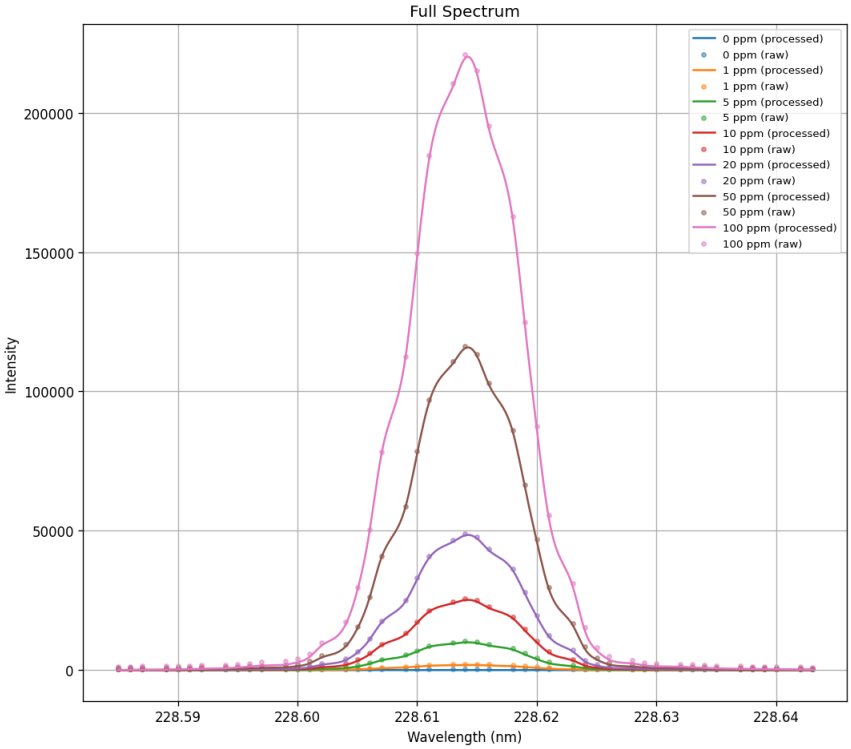
Grid Arama Sonuçlarının Açıklaması:

- **Peak NRMSE (Normalized Root Mean Square Error):** Bu değer, işlenmiş spektral verilerdeki peak pozisyonlarının gerçek değerlerden sapmasını normalize edilmiş kök ortalama kare hatası olarak ölçer.
- **RMSE (Root Mean Square Error):** Spektral verilerdeki genel hata miktarını kök ortalama kare hatası olarak ifade eder. Düşük RMSE değeri, orijinal spektral bilginin büyük ölçüde korunduğunu işaret eder.
- **N (Nokta Sayısı):** İşleme sonrası elde edilen spektral veri noktalarının toplam sayısını belirtir.

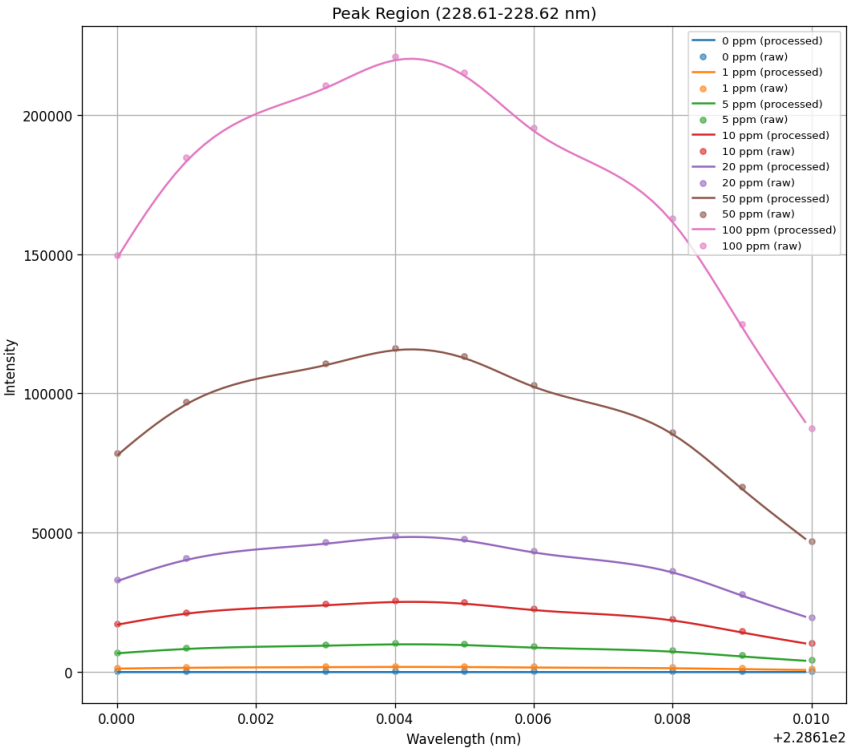
```
EN IYII AYAR (WL=3, PO=2, F=21):
  Peak NRMSE: 0.002772 (0.277%) - (dusuk olmali)
  RMSE:       611.1714 - (dusuk olmali)
  N:          63 - (yuksek olmali)
```

Listing 20: En İyi Ayar Sonuçları

1.4.6 En İyi Değerler ile Co'nun Yeni Pik Bölgesinin Grafiği



Şekil 5: Co'nun Pik Bölgesinin Detaylı Grafiği



Şekil 6: Co'nun Pik Bölgesinin Detaylı Grafiği

1.4.7 Hybrid-2Seg

Şuana kadar en iyi değerleri, ham değerlerin üzerinden olabildiğince yakın geçen ve tepe değerine olabildiğince yakın olacak şekilde; tek bir WL ve PO değerlerinde bir model ile aldık.

Şimdi ise 2 farklı yaklaşım denenip sonuçları karşılaştırılacak. Bu iki farklı yaklaşım, spektrum verilerini tek almak yerine parçalara bölerek her parçada farklı WL ve PO değerlerinin denenmesi sonucunda sonucun ve grafiğin nasıl değişebileceğini görmek için denendi.

Hybrid-2Seg

Bu yaklaşımda tepe noktasına kadar verileri bölüp, sağ ve sol kısımlar için ayrı ayrı Grid Araması çalıştırılacak. Bu özellikle tepe noktasının sağı ve solundaki eğrilerin birbirlerini etkilemeyip daha uyumlu bir eğri elde edilmesi amacıyla düşünüldü.

```
    LEFT_WLs = list(range(3, 101, 2))
    LEFT_POs = list(range(1, 8))
    LEFT_Fs  = list(range(1, 61, 1))

    RIGHT_WLs = list(range(3, 101, 2))
    RIGHT_POs = list(range(1, 8))
    RIGHT_Fs  = list(range(1, 61, 1))
```

Listing 21: Grid Arama Sonuçları, Sol-Sağ

Grid Search optimizasyonu ile hibrit spektra oluşturma işlemi başlatıldı. Spektrum verileri tepe noktası ($\lambda \approx 228.614$ nm) etrafında iki parçaya bölündü ve her parça için ayrı ayrı optimizasyon yapıldı.

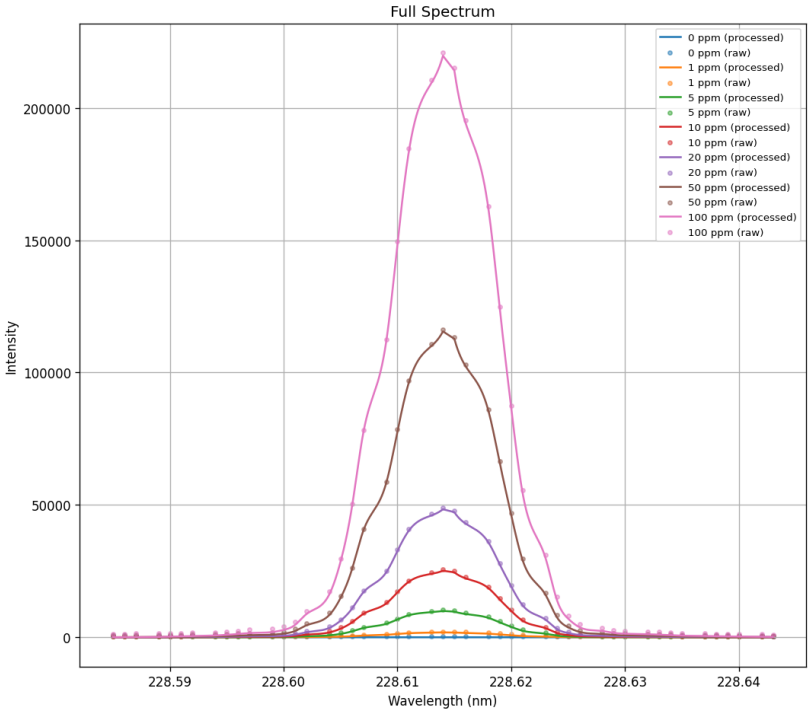
```
=== Hybrid-2Seg - Grid Search Optimizasyonu ===
Split lambda (nm) yaklasik 228.614186

1. Sol taraf (tepe oncesi) parametreleri optimize ediliyor...
Sol taraf icin grid search baslatiliyor...
Yeni en iyi sol parametreler: {'interp': 'pchip', 'F': 1, 'WL': 3, 'PO': 1, 'MED': None} (Score: 0.006270)
Ilerleme: 100/185220 (0.1%)
Ilerleme: 200/185220 (0.1%)
...
Ilerleme: 1000/185220 (0.5%)
Yeni en iyi sol parametreler: {'interp': 'pchip', 'F': 2, 'WL': 3, 'PO': 1, 'MED': None} (Score: 0.001429)
Ilerleme: 1100/185220 (0.6%)
...
```

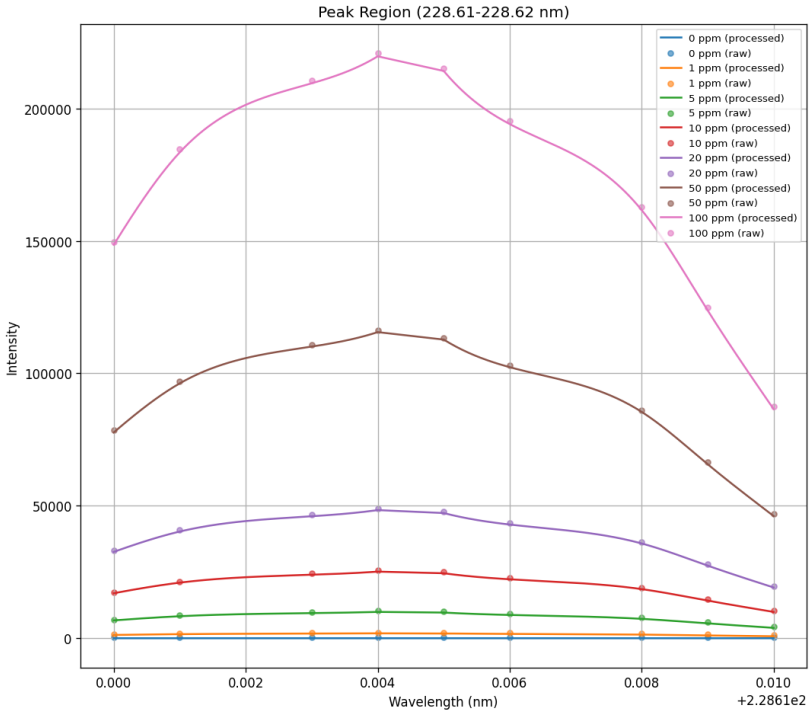
Listing 22: Grid Search Optimizasyon Süreci

```
Tamamlandi! 7 konsantrasyon icin hibrit spektra olusturuldu.
Denenen toplam kombinasyon: 185220 + 246960 = 432180 | Sure: 51m 9.6s
Sol taraf parametreleri: {'interp': 'linear', 'F': 52, 'WL': 3, 'PO': 1, 'MED': None}
Sag taraf parametreleri: {'interp': 'linear', 'F': 39, 'WL': 3, 'PO': 1, 'MED': None}
```

Listing 23: Grid Search Optimizasyon Tamamlandı



Şekil 7: Co'nun Pik Bölgesinin Detaylı Grafiği



Şekil 8: Co'nun Pik Bölgesinin Detaylı Grafiği

Optimize Edilmiş Parametreler

```
=== İİOPTMZE İİŞEDLM PARAMETRELER ===
Split Wavelength: 228.614186 nm
Sol taraf (tepe oncesi): {'interp': 'linear', 'F': 52, 'WL': 3, 'PO': 1, 'MED': None}
Sag taraf (tepe sonrasi): {'interp': 'linear', 'F': 39, 'WL': 3, 'PO': 1, 'MED': None}
```

Listing 24: Hybrid-2Seg

1.4.8 Hybrid-3Seg

Önceki yaklaşım olan Hybrid-2Seg de görüldüğü üzere, tepe noktasını "n" alırsak, n ile n+1 nokta arasını düz çizdiğini görüyoruz.

```
# Sol/sag bol
left_org = cur[cur['Wavelength'] <= split_wl]
right_org = cur[cur['Wavelength'] > split_wl]
```

Listing 25: Sol-Sağ bölme

Bundan dolayı başka bir yaklaşıp deneyip bu sefer bölge sayısını üçe çıkararak nasıl bir sonuç alınacağını deniyoruz.

```
Hybrid-3Seg          Hybrid-2Seg
***  -  n            ***  -  n
n    -  n+1
n+1  -  ***          n+1  -  ***
```

Listing 26: Sol-Sağ bölme

Bunun için parametreleri değiştiriyoruz

```
left_org = cur[cur['Wavelength'] <= peak_wl]
mid_org = cur[(cur['Wavelength'] > peak_wl) & (cur['Wavelength'] <= neighbor_wl)]
right_org = cur[cur['Wavelength'] > neighbor_wl]
```

Listing 27: Sol-Sağ bölme

Önceki Grid Search bilgilerini kullanarak sol-sağ değerlerini aynen alıp, orta değer için tekrar Grid Search çalıştırıyoruz:

```
MID_WLs = list(range(3, 101, 2))
MID_POs = list(range(1, 8))
MID_Fs = list(range(1, 61, 1))
```

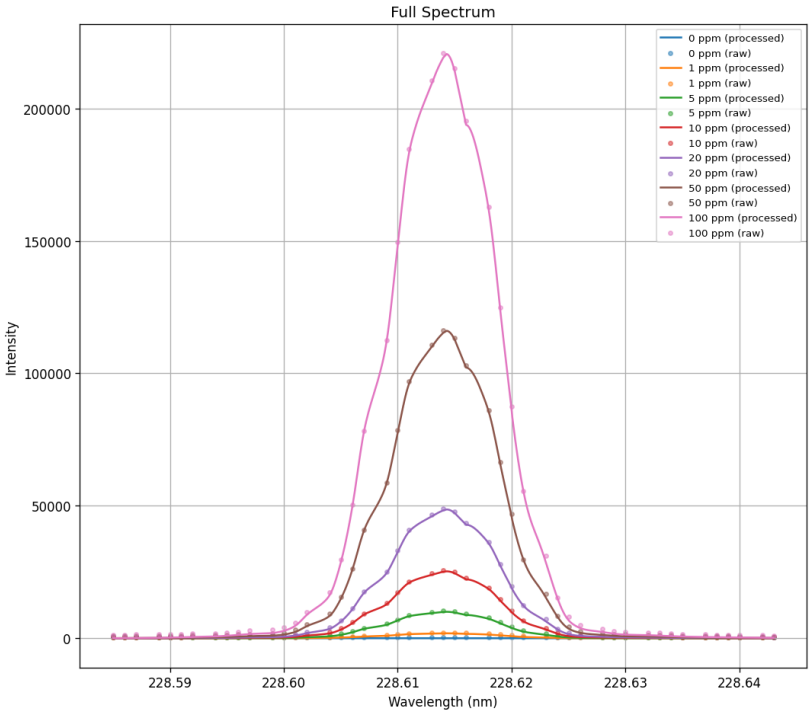
Listing 28: Orta segment için Grid Search

Sonuç olarak:

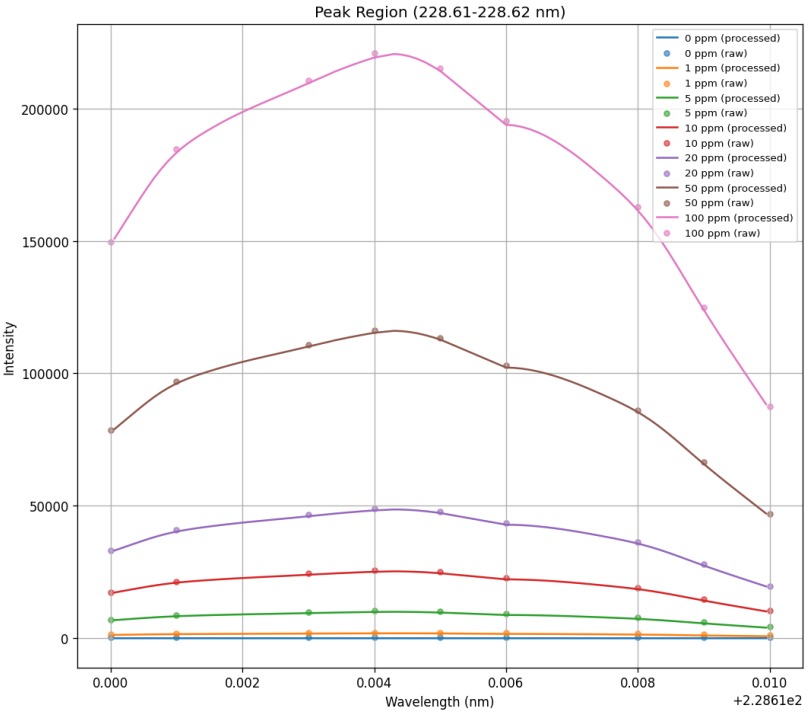
```
En iyi ORTA (→TepeSonraki): interp=akima WL=3 PO=1 F=1 MED=None | NRMSE=0.31% RMSE=678.5 N=63
Nihai 3-seg Hibrit => NRMSE=0.31% | RMSE=678.5 | N=63

    Scenario            WL_full  WL_peak    PO  Factor  Interp  Peak NRMSE  RMSE        N
0   Hybrid-3Seg-Left    13       NaN        2   12      pchip   0.003078    678.500646  63
1   Hybrid-3Seg-Mid     3        NaN        1    1      akima   0.003078    678.500646  63
2   Hybrid-3Seg-Right   31       13.0       3   10      pchip   0.003078    678.500646  63
```

Listing 29: Orta segment için Grid Search



Şekil 9: Co'nun Pik Bölgesinin Detaylı Grafiği



Şekil 10: Co'nun Pik Bölgesinin Detaylı Grafiği

1.5 Hibrit Yöntemlerin Performans Karşılaştırması

Farklı hibrit yaklaşımların performans sonuçları aşağıdaki tabloda karşılaştırılmıştır:

Yöntem	**Peak NRMSE (%)**	**RMSE**	**N**
1-Segment Hibrit	0.277	611.2	63
2-Segment Hibrit	0.266	586.7	63
3-Segment Hibrit	0.31	678.5	63

Tablo 1: Hibrit yöntemlerin performans karşılaştırması

Sonuç: Karşılaştırma sonuçları incelendiğinde, **2-segment hibrit yöntemin** en iyi performansı sergilediği görülmektedir. Bu yöntem %0.266 Peak NRMSE değeri ile en düşük hata oranını ve 586.7 RMSE değeri ile en düşük genel hatayı sağlamıştır. 1-segment hibrit yöntem ikinci sırada yer alırken, 3-segment hibrit yöntem beklenenden daha yüksek hata değerleri göstermiştir. Bu durum, aşırı segmentasyonun bazı durumlarda performansı olumsuz etkileyebileceğini göstermektedir.

2 Kod Listeleri

Listings