Water Cloud Model for Sentinel-1 Backscatter Prediction

RADAR Tutorial

2026-02-24

Table of contents

1 Overview

This tutorial demonstrates how to use the Water Cloud Model (WCM) to predict Sentinel-1 C-band backscatter over forest. We'll use observations of Leaf Area Index (LAI), Soil Water Content (SWC), and Live Fuel Moisture Content (LFMC) to model radar backscatter, then compare the model predictions with actual Sentinel-1 observations. Finally, you'll manually calibrate the model parameters to achieve the best fit between modeled and observed backscatter.

ℹ Tutorial Background and Data

Data source: Time series extracted for Tharandt forest near Dresden, Germany (50.86°N, 14.13°E)
Study site: Mixed temperate forest with continuous Sentinel-1 monitoring
Input data includes:

- Sentinel-1 backscatter in VV and VH polarizations
- Local incidence angle (LIA)
- Soil Water Content (SWC) from ERA5-Land reanalysis
- Leaf Area Index (LAI) from Proba-V and Sentinel-3 (Copernicus Global Land Service)
- Live Fuel Moisture Content (LFMC) derived from Sentinel-3

Time period: October 2018 to December 2022 (195 observations)
Data file: `py_wcm_input.csv`

❗ Important Notes on Sentinel-1 Data

When working with Sentinel-1 radar data, remember:

- **Always use RTC products** (Radiometric Terrain Corrected): These products have been radiometrically corrected to minimize terrain effects, allowing the signal to primarily reflect surface properties rather than topographic artifacts.
- **Avoid mixing orbits**: Sentinel-1 has ascending and descending orbits that observe the scene from different geometries. Combining them results in inconsistent backscatter signatures due to varying look angles and aspect-dependent scattering.

- **Speckle is inherent**: Radar images contain speckle noise. Temporal or spatial averaging can reduce this noise, though at the cost of temporal or spatial resolution.

2 Prerequisites

Import the required Python packages:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt

import warnings
warnings.filterwarnings("ignore")
```

3 Part 1: Helper Functions and Data Loading

3.1 Backscatter Conversion Functions

Radar backscatter is typically expressed in decibels (dB) or linear scale. These functions allow conversion between the two:

```
# Convert decibel to linear scale
def db_to_lin(db_value):
    """
    Convert backscatter from dB to linear scale.

    Parameters:
    -----------
    db_value : float or array
        Backscatter value in decibels

    Returns:
    --------
    float or array
        Backscatter value in linear scale
    """
    return np.power(10, (db_value / 10))

# Convert linear to decibel scale
```

```python
def lin_to_db(linear_value):
    """
    Convert backscatter from linear scale to dB.

    Parameters:
    -----------
    linear_value : float or array
        Backscatter value in linear scale

    Returns:
    --------
    float or array
        Backscatter value in decibels
    """
    return 10 * np.log10(linear_value)
```

> **Why Use Decibels?**
>
> The decibel (dB) scale compresses the large dynamic range of backscatter values into a smaller interval, making visualization and interpretation easier. Most Sentinel-1 products are provided in dB.

3.2 Loading the Dataset

Load the time series data for the Tharandt forest study site:

```python
# Load input data
df = pd.read_csv('py_wcm_input.csv', index_col=0, parse_dates=True)

# Convert LFMC from percentage to decimal (0-1)
df.LFMC = df.LFMC.mul(0.01)

# Display dataset information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 195 entries, 2018-10-12 to 2022-12-20
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
```

```
 0   VV      195 non-null    float64
 1   VH      195 non-null    float64
 2   LIA     195 non-null    float64
 3   SWC     195 non-null    float64
 4   LAI     195 non-null    float64
 5   LFMC    195 non-null    float64
dtypes: float64(6)
memory usage: 10.7 KB
```

Dataset Structure

The dataset contains 6 columns over 195 time steps:

- **VV**: Backscatter in VV polarization (dB)
- **VH**: Backscatter in VH polarization (dB)
- **LIA**: Local Incidence Angle (degrees)
- **SWC**: Soil Water Content (m^3/m^3)
- **LAI**: Leaf Area Index (m^2/m^2)
- **LFMC**: Live Fuel Moisture Content (0-1 scale)

4 Part 2: Exploring the Time Series

Before modeling, let's visualize the input variables to understand their temporal variability and relationships.

4.1 Time Series Visualization

```
fig, axes = plt.subplots(nrows=5, ncols=1, figsize=(6, 5), sharex=True)

# VV polarization
axes[0].plot(df.index, df.VV, 'o-', markersize=3, linewidth=0.5)
axes[0].set_ylabel('VV Backscatter (dB)')
axes[0].grid(True, alpha=0.3)
axes[0].set_title('Input Time Series for Water Cloud Model')

# VH polarization
axes[1].plot(df.index, df.VH, 'o-', markersize=3, linewidth=0.5, color='orange')
axes[1].set_ylabel('VH Backscatter (dB)')
axes[1].grid(True, alpha=0.3)
```

```
# LAI
axes[2].plot(df.index, df.LAI, 'o-', markersize=3, linewidth=0.5, color='green')
axes[2].set_ylabel('LAI (m²/m²)')
axes[2].grid(True, alpha=0.3)

# SWC
axes[3].plot(df.index, df.SWC, 'o-', markersize=3, linewidth=0.5, color='brown')
axes[3].set_ylabel('SWC (m³/m³)')
axes[3].grid(True, alpha=0.3)

# LFMC
axes[4].plot(df.index, df.LFMC, 'o-', markersize=3, linewidth=0.5, color='blue')
axes[4].set_ylabel('LFMC (0-1)')
axes[4].set_xlabel('Date')
axes[4].grid(True, alpha=0.3)

plt.tight_layout()
plt.show()
```

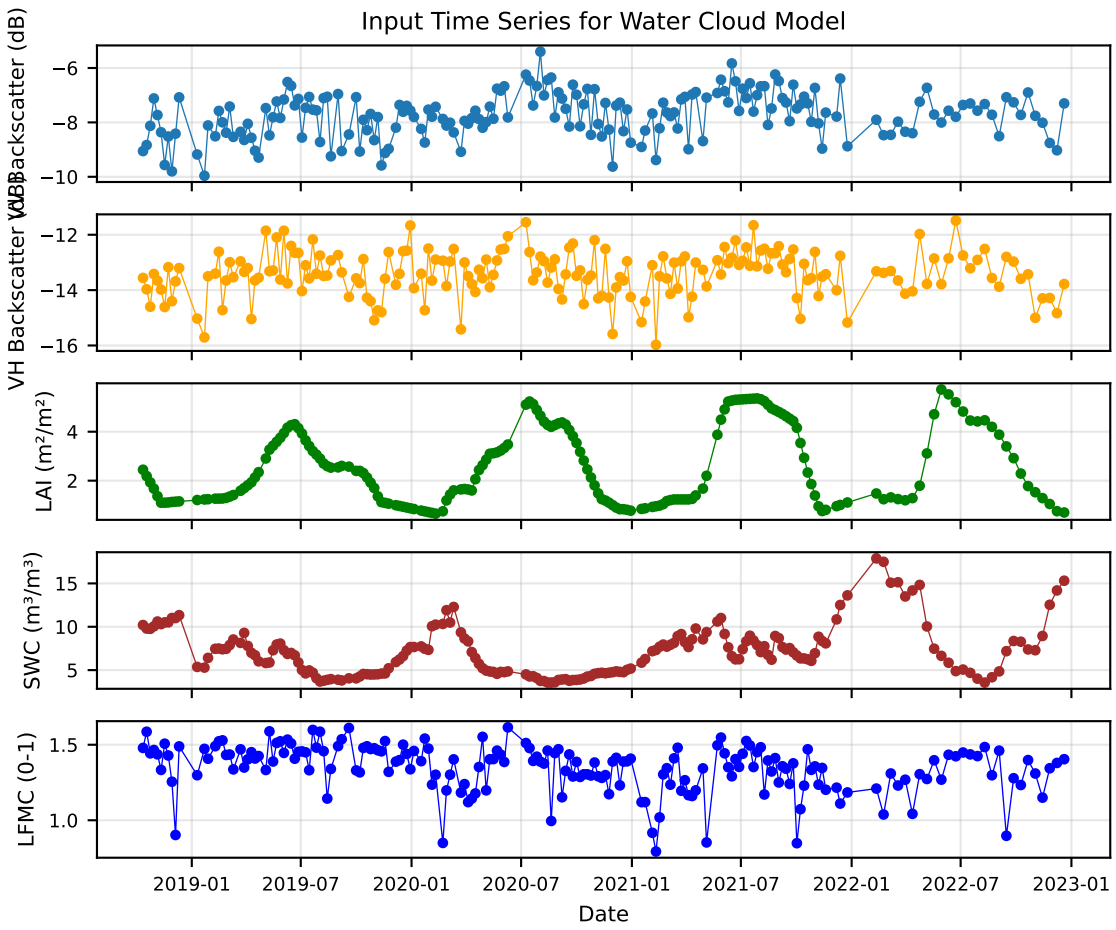


Figure 1: Time series of input variables for the Water Cloud Model

Interpreting the Time Series

Key patterns to observe:

- **Backscatter (VV, VH)**: Seasonal variability related to forest phenology, leaf-on/leaf-off conditions, and soil moisture dynamics
- **LAI**: Distinct seasonal cycle with maximum values in summer growing season and minimum during winter dormancy
- **SWC**: Reflects precipitation patterns, drought periods, and seasonal water availability
- **LFMC**: Indicates vegetation water stress and moisture content in live biomass, important for vegetation health assessment

5 Part 3: Implementing the Water Cloud Model

Now we'll implement the Water Cloud Model step by step. The model separates total backscatter into vegetation and soil components.

5.1 Step 1: VV Polarization Example

Let's start with VV polarization to understand the model implementation:

```
# Define model parameters (initial guess values)
A = 0.018  # Vegetation scattering coefficient
B = 0.09   # Vegetation attenuation coefficient
C = 0.01   # Surface roughness parameter
D = 0.001  # Soil moisture influence parameter

# Convert VV backscatter from dB to linear scale
VV_linear = db_to_lin(df.VV)

# Convert incidence angle to radians and compute cosine
LIA_rad = np.deg2rad(df.LIA)
cos_lia = np.cos(LIA_rad)

# Vegetation parameters
V1 = df.LAI * df.LFMC  # Combined vegetation descriptor
V2 = df.LAI             # Leaf Area Index

# Two-way transmissivity (attenuation through vegetation)
tau2 = np.exp(-2 * B * V2 / cos_lia)

# Vegetation backscatter component
bs_veg = A * V1 * cos_lia * (1 - tau2)

# Soil backscatter component
bs_soil = (C + D * df.SWC) * (cos_lia ** 3)

# Total backscatter (linear scale)
bs_total = bs_veg + tau2 * bs_soil

# Convert back to dB for comparison
bs_total_db = lin_to_db(bs_total)
```

Model Components

The Water Cloud Model separates backscatter into:

1. **Vegetation component** (σ_{veg}): Scattering from leaves, branches, and trunk
2. **Soil component** (σ_{soil}): Backscatter from ground surface, attenuated by vegetation
3. **Two-way transmissivity** (τ^2): Attenuation of the signal passing through vegetation

Note that temporal or spatial averaging is often necessary to reduce speckle noise and reveal meaningful patterns in Sentinel-1 time series over forests.

5.2 Visualizing Model Results

Compare the modeled backscatter with Sentinel-1 observations:

```
fig, ax = plt.subplots(figsize=(6, 3))

# Observed backscatter
ax.plot(df.index, df.VV, 'o-', label='Observed (Sentinel-1)',
        markersize=4, linewidth=1, alpha=0.7)

# Modeled backscatter
ax.plot(df.index, bs_total_db, 's-', label='Modeled (WCM)',
        markersize=4, linewidth=1, alpha=0.7)

ax.set_xlabel('Date')
ax.set_ylabel('Backscatter (dB)')
ax.set_title('Water Cloud Model vs. Sentinel-1 Observations (VV Polarization)')
ax.legend()
ax.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()
```

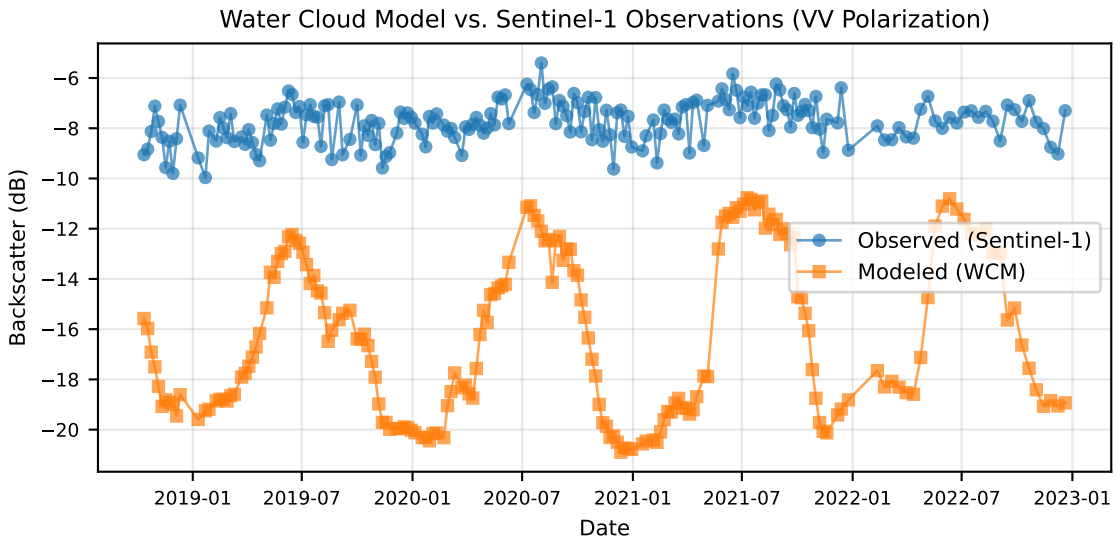


Figure 2: Comparison of modeled and observed VV backscatter

5.3 Analyzing Model Sensitivity

Examine how each input parameter influences the modeled backscatter:

```
fig, axes = plt.subplots(ncols=3, figsize=(6, 2))

# LAI vs. Backscatter
axes[0].scatter(df.LAI, bs_total_db, alpha=0.6, s=30)
axes[0].set_ylabel('$\\sigma^0_{total}$ (dB)')
axes[0].set_xlabel('LAI (m²/m²)')
axes[0].set_title('LAI Influence on Backscatter')
axes[0].grid(True, alpha=0.3)

# SWC vs. Backscatter
axes[1].scatter(df.SWC, bs_total_db, alpha=0.6, s=30, color='brown')
axes[1].set_ylabel('$\\sigma^0_{total}$ (dB)')
axes[1].set_xlabel('SWC (m³/m³)')
axes[1].set_title('Soil Moisture Influence')
axes[1].grid(True, alpha=0.3)

# LFMC vs. Backscatter
axes[2].scatter(df.LFMC, bs_total_db, alpha=0.6, s=30, color='blue')
axes[2].set_ylabel('$\\sigma^0_{total}$ (dB)')
```

```
axes[2].set_xlabel('LFMC (0-1)')
axes[2].set_title('Fuel Moisture Influence')
axes[2].grid(True, alpha=0.3)

plt.tight_layout()
plt.show()
```

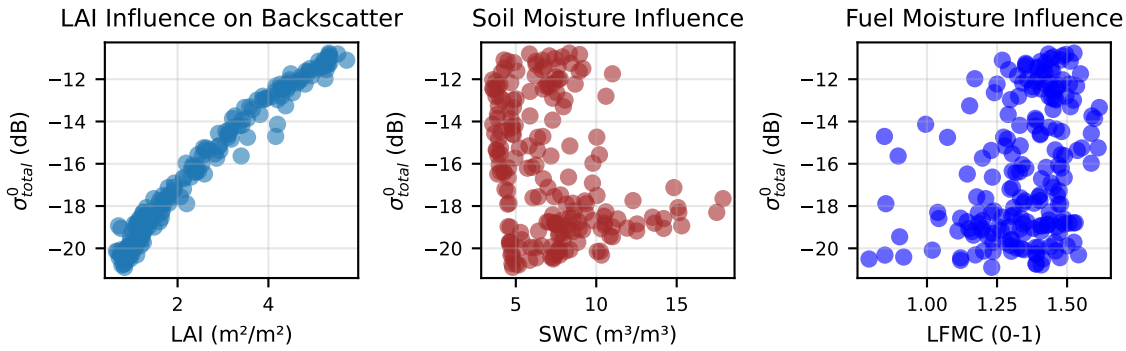


Figure 3: Relationship between modeled backscatter and vegetation parameters

6 Part 4: Manual Calibration Exercise

In this section, you will manually calibrate the Water Cloud Model for **VH polarization**, which exhibits greater sensitivity to vegetation structure compared to VV polarization.

! Your Task

Complete the following steps to calibrate the WCM for VH polarization:

1. **Write a WCM function** that combines the model code and returns total backscatter
2. **Write an RMSE function** to compute the Root Mean Squared Error between observed and predicted values
3. **Call the WCM** with initial parameter values (A, B, C, D) and visualize results
4. **Iterate the calibration** by testing different parameter values to minimize RMSE

Initial parameter values (first guess): - A = 0.018 - B = 0.09 - C = 0.01 - D = 0.001
Calibration strategy: - Vary parameters individually to isolate their effects - Test ±50% variations to assess sensitivity - Document parameter combinations that yield optimal RMSE

6.1 Task 1: Complete the WCM Function

```python
def WCM(df, A, B, C, D, polarization='VH'):
    """
    Water Cloud Model for backscatter prediction.

    Parameters:
    -----------
    df : pandas.DataFrame
        DataFrame containing input data (LAI, SWC, LFMC, LIA)
    A : float
        Vegetation scattering coefficient
    B : float
        Vegetation attenuation coefficient
    C : float
        Surface roughness parameter
    D : float
        Soil moisture influence parameter
    polarization : str
        Polarization ('VV' or 'VH')

    Returns:
    --------
    numpy.ndarray
        Modeled backscatter in dB
    """
    ### TODO: Insert your code here (adapt from the VV example above)

    # 1. Convert incidence angle
    LIA_rad = np.deg2rad(df.LIA)
    cos_lia = np.cos(LIA_rad)

    # 2. Vegetation parameters
    V1 = df.LAI * df.LFMC
    V2 = df.LAI

    # 3. Two-way transmissivity
    tau2 = np.exp(-2 * B * V2 / cos_lia)

    # 4. Vegetation component
    bs_veg = A * V1 * cos_lia * (1 - tau2)
```

```
    # 5. Soil component
    bs_soil = (C + D * df.SWC) * (cos_lia ** 3)

    # 6. Total backscatter
    bs_total = bs_veg + tau2 * bs_soil

    # 7. Convert to dB
    bs_total_db = lin_to_db(bs_total)

    return bs_total_db
```

6.2 Task 2: Write the RMSE Function

```
def rmse(actual, predicted):
    """
    Calculate Root Mean Squared Error.

    Parameters:
    -----------
    actual : array-like
        Observed values
    predicted : array-like
        Predicted values from model

    Returns:
    --------
    float
        RMSE value
    """
    ### TODO: Insert your code here
    # Formula: RMSE = sqrt(mean((actual - predicted)^2))

    mse = np.mean((actual - predicted) ** 2)
    rmse_value = np.sqrt(mse)

    return rmse_value
```

Understanding RMSE

Root Mean Squared Error (RMSE) quantifies model prediction accuracy:

- **Lower RMSE values** indicate better agreement between model predictions and observations
- RMSE is expressed in the same units as the measured variable (dB in this case)
- RMSE < 1 dB typically represents excellent model performance
- RMSE < 2 dB is generally considered acceptable

Objective: Identify parameter values that minimize RMSE for VH polarization, thereby optimizing the model fit to observed Sentinel-1 backscatter.

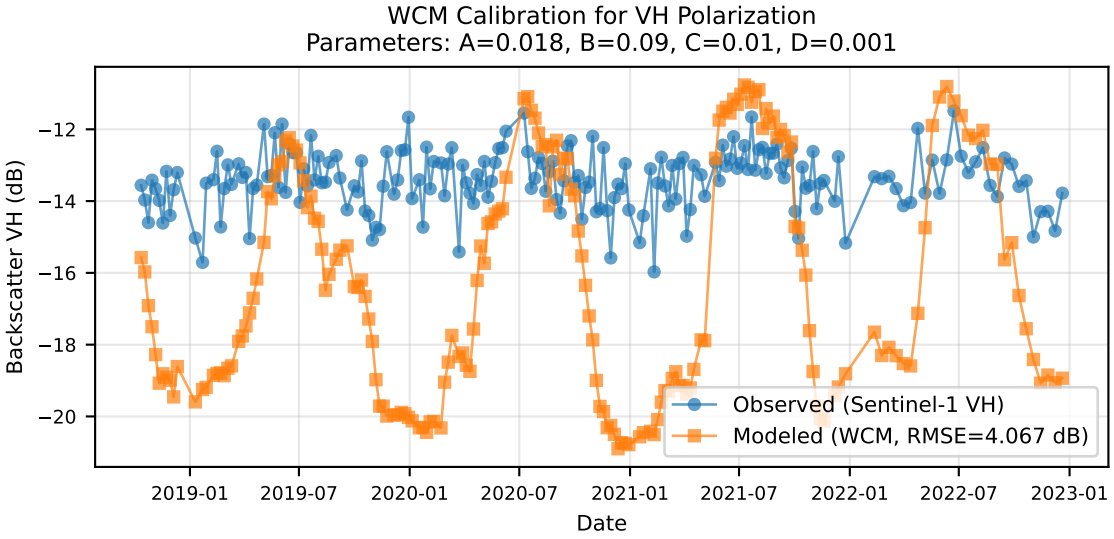
6.3 Task 3 + 4: Calibration and Visualization

Use the functions to test different parameter combinations:

```
# Initial parameter values
A = 0.018
B = 0.09
C = 0.01
D = 0.001

# Run the model for VH polarization
bs_total_vh = WCM(df, A, B, C, D, polarization='VH')

# Calculate RMSE
rmse_value = rmse(df.VH, bs_total_vh)
print(f"RMSE = {rmse_value:.3f} dB")

# Visualize results
fig, ax = plt.subplots(figsize=(6, 3))

ax.plot(df.index, df.VH, 'o-', label='Observed (Sentinel-1 VH)',
        markersize=4, linewidth=1, alpha=0.7)
ax.plot(df.index, bs_total_vh, 's-',
        label=f'Modeled (WCM, RMSE={rmse_value:.3f} dB)',
        markersize=4, linewidth=1, alpha=0.7)

ax.set_xlabel('Date')
ax.set_ylabel('Backscatter VH (dB)')
```

```
ax.set_title(f'WCM Calibration for VH Polarization\n' +
             f'Parameters: A={A}, B={B}, C={C}, D={D}')
ax.legend()
ax.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()
```

```
RMSE = 4.067 dB
```



Calibration Strategy

Systematic approach to parameter calibration:

1. **Start with parameter A** (vegetation influence):
   - Increase A → higher total backscatter
   - Decrease A → lower total backscatter
2. **Move to parameter B** (vegetation attenuation):
   - Higher B → more attenuation → less soil contribution
   - Lower B → less attenuation → more soil contribution
3. **Adjust parameter C** (soil baseline):
   - Increase C → higher baseline soil backscatter

- Decrease C → lower baseline soil backscatter

4. **Fine-tune parameter D** (soil moisture sensitivity):
   - Higher D → stronger effect of SWC variability
   - Lower D → weaker effect of SWC variability

Practical tips:

- Change parameters one at a time
- Try ±50% of the initial value first
- Observe both RMSE and visual fit to the time series
- Record your best parameter combination

6.4 Parameter Experimentation

Test different parameter values systematically:

```
# Example: Testing different values of parameter A
A_values = [0.009, 0.018, 0.027, 0.036]  # -50%, original, +50%, +100%

print("Testing parameter A:")
print("-" * 40)
for A_test in A_values:
    bs_test = WCM(df, A_test, B, C, D, polarization='VH')
    rmse_test = rmse(df.VH, bs_test)
    print(f"A = {A_test:.4f}, RMSE = {rmse_test:.3f} dB")

# TODO: Repeat for parameters B, C, D
# TODO: Find the best combination of all parameters
# TODO: Record your final parameters and RMSE
```

7 Part 5: Real-World Context - Forest Fire Example

Case Study: Czech National Park Fire

A recent analysis examined Sentinel-1 observations following a forest fire in a Czech national park. The study revealed important insights about C-band radar limitations in forested environments:
Detection capabilities:

- Significant backscatter changes were observed in grassland areas where fire removed all vegetation
- VH polarization showed reduced backscatter in burned grassland areas, enabling change detection

Limitations in forest environments: ¨

- Residual forest structure (standing dead trunks and major branches) continued to produce polarimetric scattering after the fire
- Detectable changes were primarily confined to grassland areas rather than forest plots
- C-band wavelength has limited penetration into dense forest structure, making it challenging to detect understory changes

Implications for modeling: This case demonstrates that while the Water Cloud Model can predict backscatter over intact forests, disturbances such as fire may leave structural elements (trunks, branches) that continue to influence the radar signal. Complete removal of vegetation, as occurs in grasslands, produces more detectable changes in C-band observations.
Study site characteristics: The affected area now exhibits distinct zones: healthy forest, bark beetle-damaged forest, and fire-affected areas, providing a natural laboratory for studying radar responses to different disturbance types.

Accessing Sentinel-1 Data

Modern cloud-based platforms enable efficient radar data access and processing without requiring local data downloads:
Cloud-based platforms:

- Microsoft Planetary Computer - Direct API access to analysis-ready data
- Copernicus Data Space - Official ESA data hub
- Alaska Satellite Facility - Specialized SAR archive
- Google Earth Engine - Cloud-based geospatial processing

Key advantages:

- Direct access to pre-processed RTC products
- Server-side processing capabilities
- No local storage constraints
- Streamlined time series analysis workflows

Critical requirement: Always utilize RTC (Radiometric Terrain Corrected) products to minimize terrain-induced backscatter variations and ensure measurements primarily reflect surface properties and vegetation characteristics.

8 Summary and Exercises

9 Key Takeaways

Water Cloud Model:

- Semi-empirical model separating vegetation and soil backscatter contributions
- Requires calibration of four parameters (A, B, C, D) for specific vegetation types
- Works better for some vegetation types than others

Sentinel-1 Characteristics:

- **VH polarization** is more sensitive to vegetation structure than VV
- **RTC products** are essential for reliable results
- **Speckle** can be reduced by temporal/spatial averaging
- **Don't mix orbits** - ascending and descending see different aspects

Parameter Influences: - **LAI**: Affects amount of scattering elements - **LFMC**: Influences dielectric properties and backscatter strength - **SWC**: Controls soil component backscatter

10 Best Practices

When working with Sentinel-1:

- Always verify you're using RTC products to ensure terrain effects are minimized
- Avoid combining ascending and descending orbits due to different observation geometries
- Apply median or mean filtering to reduce speckle in time series analysis
- Validate model outputs with independent measurements when available
- Consider temporal or spatial aggregation to improve signal-to-noise ratio in forested areas

When calibrating WCM:

- Initialize with published parameter values for similar vegetation types and conditions
- Employ systematic calibration approaches, varying parameters individually
- Utilize independent validation datasets to assess model generalizability
- Maintain comprehensive documentation of calibration procedures and results
- Recognize that temporal or spatial averaging may be necessary to reduce noise and reveal underlying patterns in dense vegetation

11 Exercises

1. Polarization Comparison

- Calibrate WCM for both VV and VH polarizations
- Compare optimal parameters and RMSE
- Which polarization fits the model better?

2. Seasonal Analysis

- Split data into winter and summer seasons
- Calibrate separately for each season
- Do optimal parameters differ between seasons?

3. Sensitivity Analysis

- Systematically vary each parameter in small steps
- Create plots of RMSE vs. parameter value
- Which parameter has the strongest influence on model fit?

4. Model Simplification

- Try simplified model versions (e.g., constant SWC)
- How much does RMSE increase?
- Is the full model complexity necessary?

5. Cross-Validation

- Split data into calibration (70%) and validation (30%) sets
- Calibrate on training data, test on validation data
- Does the model generalize well?

12 Conclusion

The Water Cloud Model provides a physically-based framework for understanding and predicting radar backscatter over vegetated surfaces. While it requires careful calibration and has limitations, it offers valuable insights into:

- Temporal dynamics of vegetation and soil moisture
- Relationships between biophysical parameters and radar observations
- Detection of vegetation stress and changes
- Soil moisture monitoring beneath vegetation canopy

Success with WCM requires understanding its limitations, careful calibration, and validation with independent data. As demonstrated by the forest fire example, C-band Sentinel-1 can detect some types of land surface changes but may struggle with subtle changes in dense forest structure.

⚠ Important Considerations

Model Limitations:

- C-band (Sentinel-1) has limited forest penetration
- Model assumes homogeneous vegetation within pixel
- Parameters may vary seasonally and with vegetation type
- Structural elements (trunks, large branches) persist even after disturbances

Data Quality:

- Use only RTC products
- Check for temporal decorrelation
- Be aware of speckle effects
- Consider multi-temporal averaging for stable estimates

This tutorial provides hands-on experience with radar backscatter modeling. The Water Cloud Model serves as a foundation for understanding radar-vegetation interactions and can be extended with more sophisticated approaches for specific applications.