Time-Domain Forecasting and Frequency-Domain Analysis for Solar Irradiance Data

Amber Rexwinkle[1] Justin Nichols[2] Alexis Denhard[3]

Colorado School of Mines
AMS Department

Abstract

Solar irradiance forecast are essential to integrating solar power into the electric grid; however, may studies only explore time-domain approaches for modelling such data. Due to the diurnal cycle of solar irradiance data, natural short-term frequencies exist that should be taken into account during modelling. This project aims to determine the best short-term forecasting model use both time-domain and frequency-domain forecasting ARIMA models in order to determine the best model in terms of prediction accuracy. This project concludes that differenced ARIMA models for a lag of 29 perform best for forecasting in terms of statistical metrics compared to standard time-domain ARIMA models. This finding implies that standard ARIMA models do not fully capture the short-term frequencies present in solar irradiance data and benefit from a frequency-domain differencing component.

[1]arexwinkle@mines.edu
[2]jnichols2@mines.edu
[3]adenhard@mines.edu

Contents

List of Figures

List of Tables

1 Introduction

Solar irradiance data is dominated by the durinal cycle, experiencing frequencies corresponding with the daily rotation of the Earth. Solar irradiance forecasting is crucial for solar energy's integration into the power grid. However, many studies only approach short-term forecasting from a time-domain approach. This project aims to determine the best-performing approach for short-term horizon forecasting on direct normal irradiance (DNI) utilizing both standard time-domain forecasting autoregressive integrated moving average (ARIMA) models and ARIMA models created through frequency-domain analysis.

2 Related Works

In "Forecasting solar irradiance at short horizons: Frequency and time domain models", authors Reikard and Hansen tested several time series models for forecasting solar irradiance data over horizons of 15, 30, 45 minutes, and 1-3 hours. The methods included persistence, regressions in levels, ARIMAs, and a frequency domain model, based on Fourier transformation of the ARIMA [1]. The study found that model achievement depended on horizon length with the ARIMA and frequency domain models performing similarly when forecasting 3 hours [1]. In "Short-Term Load Forecasting Based on Frequency Domain Decomposition and Deep Learning", the development of a short-term load forecasting model was done through frequency domain decomposition inputted into a long-short term memory (LSTM) neural network [3]. The presented LSTM configuration outperformed other LSTM models in terms of statistical metrics such as mean absolute percentage error (MAPE) and root-mean squared error (RMSE). These publications motivate the short-term forecasting and frequency-domain analysis approaches for solar irradiance data.

3 Data Description and Pre-Processing

The data utilized in this project is publicly available through the National Solar Radiation Database (NSRDB) and is provided by the National Renewable Energy Laboratory (NREL). The data set consists of solar irradiance measurements as well as other meteorological parameters for eight sites across the continental United States. The measurements are recorded in 30-minute intervals. This projects focuses on the data for the site in Sioux Falls, South Dakota (SXF site) using a univariate time series of direct normal irradiance (DNI).

To pre-process the data, we first converted the time stamps provided into coordinated universal time and then America/Chicago local time corresponding to the location of the SXF site. Next, we removed night-time observations using the condition of removal of a time-stamped DNI if its corresponding solar zenith angle is greater than 89.5 degrees. There are no recorded DNI measurements for night time-stamps as the measurement depends on radiation from the sun. Lastly, we created the DNI time series object as necessary for ARIMA model fitting.

4 Short-term Prediction Horizons

In fitting the models, we wanted to see if there was an impact in predictive power based on two time windows of the data- the short term forecasting horizons and the fitted observations leading up to the forecast horizons. Each method forecasts three short-term horizons: 4, 8, and 12-hours ahead, corresponding to 8, 16, and 24 prediction steps ahead (since the data is measured in thirty-minute intervals as aforementioned). Additionally, each method is trained on three subsets of data leading up to the prediction period: 3, 5, and 7 days before the prediction date. Note that as we removed night-time observations the time stamps do not sequentially align. For example, the DNI time series does not contain hours 2–3 as the solar zenith angle is greater than 89.5 degrees.

Though we removed the night-time measurements of DNI, we still treat the time series as a continuous and complete data set with each time step corresponding to hours after the pre-processing of the data rather than the recorded timestamp. To mitigate weather and cloud issues that directly affect DNI measurements, we chose to predict the summer day, June 8^{th}, 2019. Therefore, our subsets of training days are June 1^{st}–7^{th}, June 3^{rd}–7^{th}, and June 5^{th}–7^{th}. These training subsets are shown in Figure 1 below.

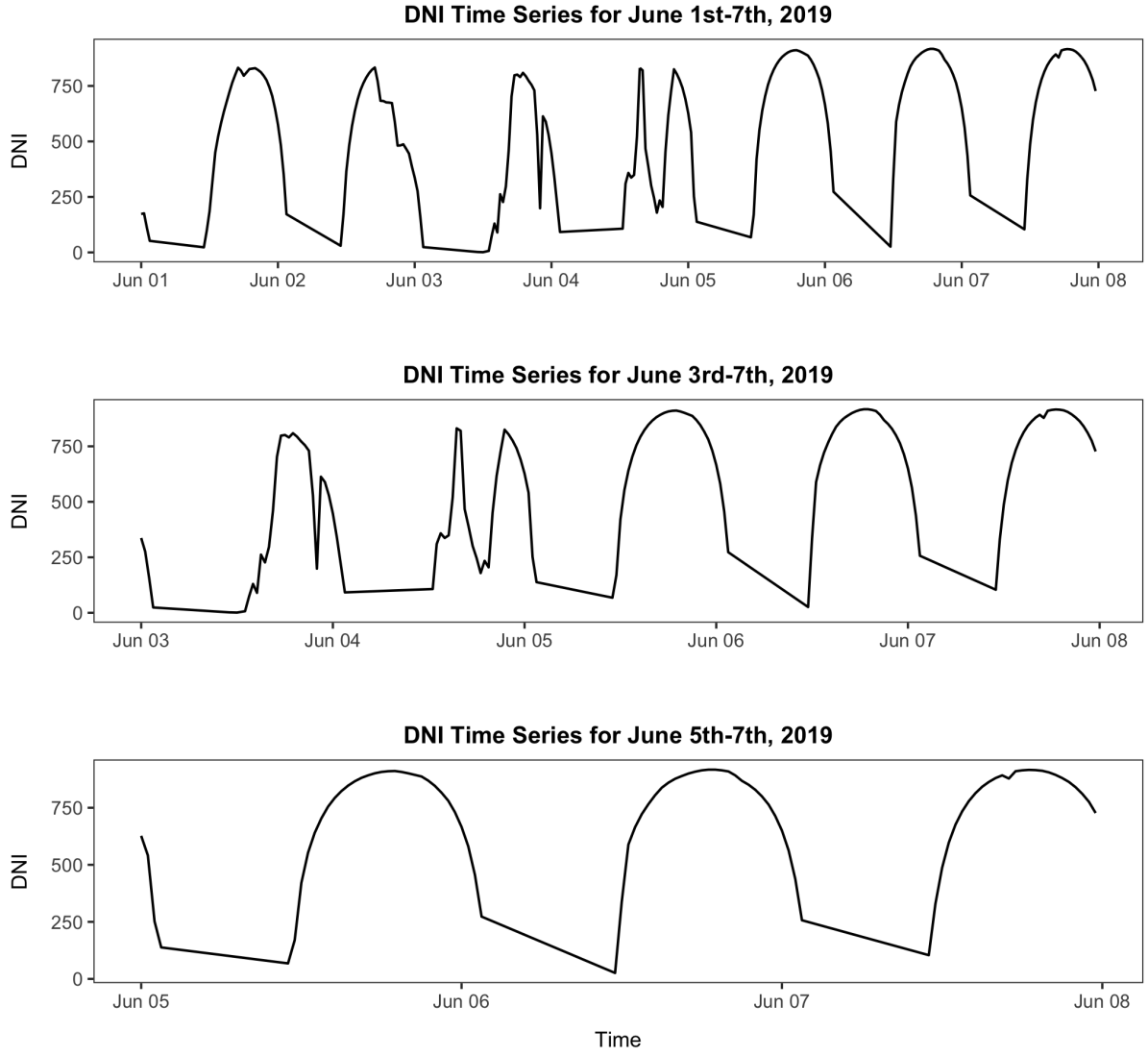


Figure 1: Subsets of Training Sets Used to fit ARIMA Models

5 Statistical Metrics

To quantify predictive performance of the models, we calculate root-mean squared error (RMSE), mean absolute percentage error (MAPE), and mean bias error (MBE). The equations for these metrics are listed below where N corresponds to the number of data, y_{true} is the actual value of DNI at time t, and y_i corresponds to the predicted DNI at time t.

5.1 Root-Mean Squared Error

RMSE measures the square root of the averaged squared error. The metric gives more important to more significant errors. RMSE is most relevant when caring about larger errors or sensitivity to outliers. We want to minimize RMSE for higher forecast accuracy.
Talk about advantages and disadvantages

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - y_{\text{TRUE}})^2}$$

5.2 Mean Absolute Percentage Error

MAPE measures the sum of individual absolute errors divided by the total number of observations. It is the average of the percentage errors. This metric experiences skew with high errors during low periods significantly affecting the measurement. Due to this, we must use caution in interpreting MAPE. We want to minimize MAPE for higher forecast accuracy.

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}|y_{\text{TRUE}} - y_i|$$

5.3 Mean Bias Error

MBE measures the averages sum of differences between predictions and actual observations. The metric is more robust to outliers. Caution must be used as positive and negative errors can cancel each other out. For MBE, we look for values closest to zero for higher model forecast accuracy.

$$\text{MBE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - y_{\text{TRUE}})$$

6 Time-Domain Model

The ARIMA models were fit on each of the three subsets of training days using the auto.arima function, where the model returned has the smallest AIC value. The function conducts a search over all possible models within the order constraints provided. For the 3 models, we ended up with the following parameters:

Table 1: ARIMA Models for 3 time horizons

June 1st–7th, 2019	ARIMA(0,1,1)
June 3rd–7th, 2019	ARIMA(2,1,1)
June 5th–7th, 2019	ARIMA(2,0,0)

The aforementioned models were used to forecast the DNI for three short-term horizons: 4, 8, and 12 hours. We then calculated the statistical metrics to determine the model with the highest prediction accuracy. The corresponding graphs of prediction and predictions intervals can be found in section B.1 of the Appendix.

Table 2: Statistical Metrics for Time-Domain Auto Arima

	Hours (Steps) Ahead	MBE	RMSE	MAPE
June 1st–7th, 2019	4 hours (8)	286.1	336.3	286.2
	8 hours (16)	106.1	248.5	187.8
	12 hours (24)	145.6	252.2	202.3
June 3rd–7th, 2019	4 hours (8)	349.1	298.9	298.9
	8 hours (16)	131.2	252.9	184.0
	12 hours (24)	175.5	267.5	210.7
June 5th–7th, 2019	4 hours (8)	272.0	323.3	272.0
	8 hours (16)	107.6	237.2	175.6
	12 hours (24)	151.8	249.7	198.1

Best Model according to specified metric Best Model for each time horizon

From the table above, we observe that the 7 day trained ARIMA model has the overall lowest MBE for a forecast interval of 8 hours compared to all other models. The 3 day trained ARIMA model for a forecast of 8 hours has the overall lowest RMSE and MAPE metrics. These values are highlighted in green. Among each training period, the best-performing metrics are highlighted in green. All training subsets have the lowest metrics for forecasting 8 hours ahead. Overall, the best model in terms of prediction accuracy is ARIMA(2, 0, 0) using a 3 day training period for forecasting 8 hours ahead.

7 Frequency-Domain Model

ARIMA models are adept at modeling the overall trend of time series with seasonal patterns, but often fall short on forecasting values that may fall outside of what is expected. Because of this, we chose to use a frequency domain approach to identify any dominant periods within the series that may improve the ARIMA model. Since DNI is only meaningful during the day, we were left with a non-consecutive series, with all night time values removed from the time series. This aids in improving the model as the 30 minute time steps causes a "long term" component to be considered measurements throughout a day. In the next section we plot the periodogram, and identify the corresponding frequency and thus, period to consider using for a difference.

7.1 Periodograms

In the plot below we have the scaled-periodogram symmetric about 1/2. We have marked the largest associated frequency with a red dot for each of the individual time series.

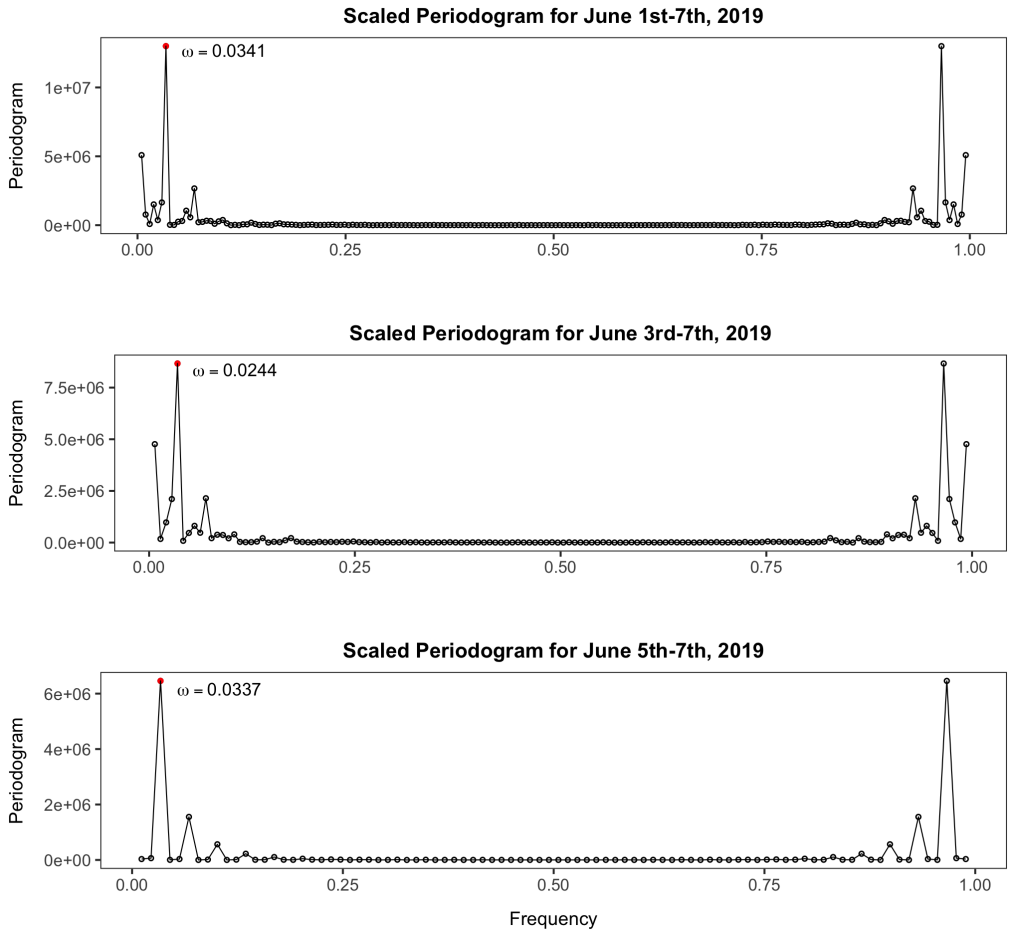


Figure 2: Periodogram of given subset with ω value noted

We use the ω value, which is pretty similar for all three models to determine a T value, which is approximately equal to $T = \frac{1}{\omega}$. Doing this calculation yields approximately 29, which is what we will consider the length of the short term lag. From here, we will difference the time series by 29 and repeat the ARIMA analysis that was done above.

7.2 Plots of Differenced Time Series

After differencing the time series we are left with a resulting time series where 29 degrees of freedom are lost. The plots of these are to follow

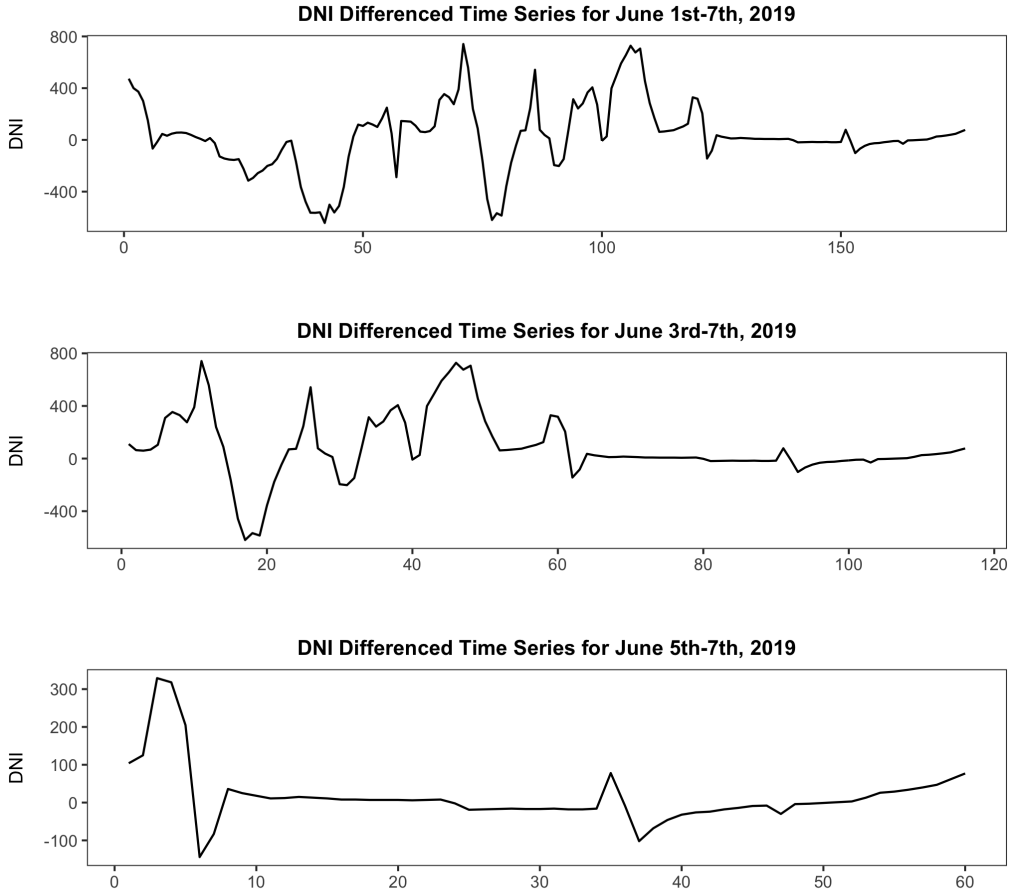


Figure 3: Subsets of Training Sets Used to fit ARIMA Models (differenced)

If we compare the overall shape or variance of these, it smoother than our original plots from Figure 1. At this point, we follow the same routine as we did in the previous section, where we will first determine the best ARIMA model.

7.3 Forecasting Results

With the difference applied to the time series, we once again determine which ARIMA model would be most suitable.

Table 3: ARIMA Models for 3 time horizons (deseasonalized)

June 1st–7th, 2019	ARIMA(1,0,1)
June 3rd–7th, 2019	ARIMA(1,0,1)
June 5th–7th, 2019	ARIMA (3,1,0)

From here, as we did with the time series before deseasonalizing it, we forecast on the 3 different time steps, and evaluate the prediction accuracy based off the same metrics. These can be seen below.

Table 4: Statistical Metrics for Differenced Time-Domain Auto Arima

	Hours (Steps) Ahead	MBE	RMSE	MAPE
June 1st–7th, 2019 (differenced)	4 hours (8)	115.0	124.7	115.0
	8 hours (16)	210.0	269.1	210.5
	12 hours (24)	290.1	342.5	290.1
June 3rd–7th, 2019 (differenced)	4 hours (8)	109.7	120.5	109.7
	8 hours (16)	131.2	252.9	184.0
	12 hours (24)	286.7	340.5	286.7
June 5th–7th, 2019 (differenced)	4 hours (8)	136.5	140.3	136.5
	8 hours (16)	247.6	304.5	247.6
	12 hours (24)	136.5	140.3	136.5

Best Model according to specified metric Best Model for each time horizon

According to these results, we see that overall no matter the subset, predictions are better done for shorter time-spans. The superior of these results being the 4-hour forecast done on the 5-day set, where we see a tie with the MBE and MAPE at 109.7. On the other hand, there is a strangeness to the result for the 3-days, 12-hour findings. In addition to the metric results, plots of the forecast can be found in Appendix B.2

8 Comparison

In order to determine which type of model preforms the best forecasting, we will compare the MBE, RMSE, and MAE between the ARIMA model and differenced ARIMA.

Table 5: Statistical Metrics for both Time-Domain Auto ARIMAs

	Hours (Steps) Ahead	MBE	RMSE	MAE
June 1^{st}–7^{th}, 2019	4 hours (8)	286.1	336.3	286.2
	8 hours (16)	106.1	248.5	187.8
	12 hours (24)	145.6	252.2	202.3
June 1^{st}–7^{th}, 2019 (differenced)	4 hours (8)	115.0	124.7	115.0
	8 hours (16)	210.0	269.1	210.5
	12 hours (24)	290.1	342.5	290.1

Table 6: Statistical Metrics for both Time-Domain Auto Arimas

	Hours (Steps) Ahead	MBE	RMSE	MAE
June 3^{rd}–7^{th}, 2019	4 hours (8)	349.1	298.9	298.9
	8 hours (16)	131.2	252.9	184.0
	12 hours (24)	175.5	267.5	210.7
June 3^{rd}–7^{th}, 2019 (differenced)	4 hours (8)	109.7	120.5	109.7
	8 hours (16)	131.2	252.9	184.0
	12 hours (24)	286.7	340.5	286.7

Table 7: Statistical Metrics for both Time-Domain Auto Arimas

	Hours (Steps) Ahead	MBE	RMSE	MAE
June 5^{th}–7^{th}, 2019	4 hours (8)	272.0	323.3	272.0
	8 hours (16)	107.6	237.2	175.6
	12 hours (24)	151.8	249.7	198.1
June 5^{th}–7^{th}, 2019 (differenced)	4 hours (8)	136.5	140.3	136.5
	8 hours (16)	247.6	304.5	247.6
	12 hours (24)	136.5	140.3	136.5

As described above, we fit ARIMA models, both standard and difference – from the frequency-domain analysis – on subsets June 1st-7th, June 3rd-7th, and June 5th-7th. We then forecasted on the models for the first 4, 8, and 12 hours on June 8th. These forecast horizons correspond to 8, 18, and 24 time steps ahead. From our analysis, we conclude that the overall best model in terms of the statistical metrics is the ARIMA(1,0,1) differenced model trained on 5 days and forecasting 4 hours ahead. Of the standard (non-differenced) ARIMA models, an ARIMA(2,0,0) model trained on 3 days and forecasting 8 hours gave the highest prediction accuracy.

We note that the differenced ARIMA models outperformed the standard ARIMA models in several cases:

- 7 days, 4 hour forecast
- 5 days, 4 and 8 hour forecasts
- 3 days, 4 and 12 hour forecasts

Of the 18 ARIMA models (6 models $\times$ 3 training subsets, 9 standard ARIMA, 9 differenced ARIMA), the differenced models outperformed the standard models for 6 of the trained subsets. Therefore, we conclude the differenced ARIMA models has higher prediction accuracy than the standard models for 2/3 of the forecasts.

9 Conclusion

This goal of this project is to forecast short-term horizons of 4, 8, and 12 hours, corresponding to 8, 16, and 24 steps ahead for the DNI time series. This data set is from Sioux Falls, South Dakota and is publicly available through the NSRDB. Due to the volatile nature of solar irradiance data from extraneous factors such as extreme weather conditions and cloud coverage, we forecasted on a summer day, June 8th, 2019 using standard and frequency-domain differenced ARIMA models. We also considered several training windows: June 1st-7th, June 3rd-7th, and June 5th-7th. Forecasting accuracy depended on both training period and forecast horizon as well as a standard versus differenced ARIMA model. Overall, the best-performing model in terms of RMSE, MAPE, and MBE was a differenced ARIMA(1,0,1) model trained on 5 days (June 3rd-June 7th) and forecasting 4 hours ahead. Among the standard ARIMA models, an ARIMA(2,0,0) model trained on 3 days (June 5th-7th) and forecasting 8 hours ahead had the highest prediction accuracy. Overall, the differenced ARIMA models had the best metrics 2/3 of the forecasts compared to the standard ARIMA models. However, due to the changing variance of the DNI series, we believe other time series models could obtain better forecasting metrics. We conclude that for several training days and short-term prediction horizons, a differenced ARIMA model is most appropriate. Frequency-domain analysis allowed for the discovery and modelling of short-term seasonality within the DNI series.

10 Future Extensions

To further fit and forecast on the DNI series, we consider using other models such as an autoregressive conditional heteroskedasticity (ARCH) model that explicitly models the change in variance over time for a time series. As seen in the analysis above, the DNI series does not exhibit constant variance with higher measurements during high sun radiation exposure and lower values during sunrise and sunset. While the ARIMA differenced model through frequency-domain analysis performed better in terms of the metrics than most of the standard ARIMA models, an ARIMA model may not be most appropriate in capturing the nature of the DNI series. Other models must be fit and forecast to optimize DNI prediction.

Appendix

A Code

A.1 Time-Domain Forecasting

Load in Data, Pre-Process, Create DNI Series and Train/Test Split

```
rm(list = ls()) #clear environment

#load in necessary libraries
library(astsa)
library(imputeTS)
library(forecast)
library(fields)
library(lubridate) #yday function
library(iemisc) #cosd function
library(purrr) #detect_index function
library(stringr)

# Load in data (SXF), pre-processing, and create DNI time series

# load in csv file for SXF (20 years) data
SXF = suppressWarnings(read.csv("D:/Work/NSRDB_all_years/SXF _43.73-96.62_all.csv"
                                , header = TRUE, sep = ","))

head(SXF)

SXF$i..time_index = str_replace(SXF$i..time_index, "T", " ")

head(SXF)

SXF$i..time_index = substr(SXF$i..time_index, 1, 19)

head(SXF)

# Change UTC to local time for SXF: lon -96.62328 and lat 43.73403
SXF$date = as.POSIXct(SXF$i..time_index, tz = "UTC", format = "%Y-%m-%d %H:%M:%S")

SXF$date[1]

# Look up time zone
tz = lutz::tz_lookup_coords(43.73, -96.2, method="accurate")
# SXF is in America/Chicago time
# tz

# Covert UTC to America/Chicago time zone
SXF$date = force_tz(SXF$date, tz = "America/Chicago")

SXF$date[1:5]

# Remove night-time observations
SXF = SXF[!(SXF$solar_zenith_angle > 89.5),]

# Remove remaining zeroes in the data (we should not have any for day time)
SXF = SXF[!(SXF$dni <= 0),]

# Create DNI time series
dni = data.frame(SXF$date, SXF$dni)
head(dni)

View(dni)
length(dni$SXF.dni)
```

Fit Standard ARIMA Models and Forecast

```
# fi arima models
ARIMA.7 = auto.arima(ts(dni_fit.7$SXF.dni))
ARIMA.5 = auto.arima(ts(dni_fit.5$SXF.dni))
ARIMA.3 = auto.arima(ts(dni_fit.3$SXF.dni))

checkresiduals(ARIMA.7)

checkresiduals(ARIMA.5)

checkresiduals(ARIMA.3)

# 4-hr pred
pred.7.4hr = predict(ARIMA.7, n.ahead = 8)
pred.5.4hr = predict(ARIMA.5, n.ahead = 8)
pred.3.4hr = predict(ARIMA.3, n.ahead = 8)

# 8-hr pred
pred.7.8hr = predict(ARIMA.7, n.ahead = 16)
pred.5.8hr = predict(ARIMA.5, n.ahead = 16)
pred.3.8hr = predict(ARIMA.3, n.ahead = 16)

# 12-hr pred
pred.7.12hr = predict(ARIMA.7, n.ahead = 24)
pred.5.12hr = predict(ARIMA.5, n.ahead = 24)
pred.3.12hr = predict(ARIMA.3, n.ahead = 24)

# 4-hour prediction errors

MBE.7.4hr = mean(pred.7.4hr$pred - dni_for.4hr$SXF.dni)
RMSE.7.4hr = sqrt(mean((pred.7.4hr$pred - dni_for.4hr$SXF.dni)^2))
MAE.7.4hr = mean(abs(pred.7.4hr$pred - dni_for.4hr$SXF.dni))

MBE.5.4hr = mean(pred.5.4hr$pred - dni_for.4hr$SXF.dni)
RMSE.5.4hr = sqrt(mean((pred.5.4hr$pred - dni_for.4hr$SXF.dni)^2))
MAE.5.4hr = mean(abs(pred.5.4hr$pred - dni_for.4hr$SXF.dni))

MBE.3.4hr = mean(pred.3.4hr$pred - dni_for.4hr$SXF.dni)
RMSE.3.4hr = sqrt(mean((pred.3.4hr$pred - dni_for.4hr$SXF.dni)^2))
MAE.3.4hr = mean(abs(pred.3.4hr$pred - dni_for.4hr$SXF.dni))

# 8-hour prediction errors

MBE.7.8hr = mean(pred.7.8hr$pred - dni_for.8hr$SXF.dni)
RMSE.7.8hr = sqrt(mean((pred.7.8hr$pred - dni_for.8hr$SXF.dni)^2))
MAE.7.8hr = mean(abs(pred.7.8hr$pred - dni_for.8hr$SXF.dni))

MBE.5.8hr = mean(pred.5.8hr$pred - dni_for.8hr$SXF.dni)
RMSE.5.8hr = sqrt(mean((pred.5.8hr$pred - dni_for.8hr$SXF.dni)^2))
MAE.5.8hr = mean(abs(pred.5.8hr$pred - dni_for.8hr$SXF.dni))

MBE.3.8hr = mean(pred.3.8hr$pred - dni_for.8hr$SXF.dni)
RMSE.3.8hr = sqrt(mean((pred.3.8hr$pred - dni_for.8hr$SXF.dni)^2))
MAE.3.8hr = mean(abs(pred.3.8hr$pred - dni_for.8hr$SXF.dni))

# 12-hour prediction errors
```

A.2 Frequency Analysis

tik

Read-in, subset, and Plot each Time Series

```
rm(list=ls())
library(tidyverse); library(gridExtra); library(gridExtra); library(ggfortify);
library(lubridate)
library(GGally); library(lemon); library(ggrepel)
library(scales);
library(latex2exp); library(scales)
library(qqplotr)
library(astsa); library(forecast)
library(reshape2)
library(magrittr)
library(grid)
library(lutz)
th <- theme_bw(base_size=10) +
  theme(panel.grid = element_blank()) +
  theme(axis.text=element_text(size=8), axis.title=element_text(size=9)) +
  theme(strip.background=element_blank()) +
  theme(strip.text=element_text(face="bold")) +
  theme(plot.margin=margin(10,5,5,5)) +
  theme(plot.title=element_text(size=10, face='bold', hjust=0.5)) +
  theme(legend.title=element_text(size=7), legend.text=element_text(size=6),
        legend.margin=margin(0,0,0,0)) +
  theme(axis.title.x = element_text(margin = unit(c(3, 0, 0, 0), "mm")),
        axis.title.y = element_text(margin = unit(c(0, 3, 0, 0), "mm")))

##### Read in full SXF data set (*change accordingly*) #####
sxf_full <- read_csv("../data/sxf_full.csv")

##### Extract datetime and DNI columns and save as .rdata #####
sxf_dni <- sxf_full %>%
  filter(dni != 0, solar_zenith_angle <= 89.5) %>%
  select(time_index, dni)

dni_06_2019_1 <- filter(sxf_dni, year(time_index)==2019 &
                          month(time_index)==6 & day(time_index) <= 7)
dni_06_2019_2 <- filter(sxf_dni, year(time_index)==2019 &
                          month(time_index)==6 & day(time_index) <= 7 &
                          day(time_index) >= 3)
dni_06_2019_3<- filter(sxf_dni, year(time_index)==2019 &
                         month(time_index)==6 & day(time_index) <= 7 &
                         day(time_index) >= 5)

g1 <- ggplot(dni_06_2019_1, aes(x=time_index, y=dni)) +
  geom_line() +
  th + labs(x='', y='DNI', title='DNI Time Series for June 1st-7th, 2019') +
  scale_x_datetime(date_break="1 day", date_labels="%b %d")
g2 <- ggplot(dni_06_2019_2, aes(x=time_index, y=dni)) +
  geom_line() +
  th + labs(x='', y='DNI', title='DNI Time Series for June 3rd-7th, 2019') +
  scale_x_datetime(date_break="1 day", date_labels="%b %d")
g3 <- ggplot(dni_06_2019_3, aes(x=time_index, y=dni)) +
  geom_line() +
  th + labs(x='Time', y='DNI', title='DNI Time Series for June 5th-7th, 2019') +
  scale_x_datetime(date_break="1 day", date_labels="%b %d")

g <- arrangeGrob(g1, g2, g3)
#ggsave('../plots/ts.png', g, bg='transparent')
```

Find Scaled-Periodogram for each and Plot

```
n <- length(dni_06_2019_1$dni)
dft <- 2*fft(dni_06_2019_1$dni)
P_dni <- Mod(dft)^2/n
P_df_1 <- subset(data.frame(x=0:(n-1)/n, y=P_dni), x>0)

n <- length(dni_06_2019_2$dni)
dft <- 2*fft(dni_06_2019_2$dni)
P_dni <- Mod(dft)^2/n
P_df_2 <- subset(data.frame(x=0:(n-1)/n, y=P_dni), x>0)

n <- length(dni_06_2019_3$dni)
dft <- 2*fft(dni_06_2019_3$dni)
P_dni <- Mod(dft)^2/n
P_df_3 <- subset(data.frame(x=0:(n-1)/n, y=P_dni), x>0)

ix_1 <- which.max(P_df_1$y)
ix_2 <- which.max(P_df_2$y)
ix_3 <- which.max(P_df_3$y)

g1 <- ggplot(P_df_1, aes(x=x, y=y)) +
  geom_point(shape=1, size=0.75) +
  geom_point(data=slice(P_df_1, ix_1), aes(x=x, y=y), color='red', size=0.75) +
  annotate('text', x=P_df_1$x[ix_1]+0.07, y=P_df_1$y[ix_1]-300000,
           label=TeX(sprintf('$\\omega = %.4f$', P_df_1$x[ix_1])), size=3) +
  geom_line(size=0.25) +
  scale_y_continuous(labels = function(x) format(x, scientific = TRUE)) +
  th + labs(x='', y='Periodogram', title='Scaled Periodogram for June 1st-7th, 2019')

g2 <- ggplot(P_df_2, aes(x=x, y=y)) +
  geom_point(shape=1, size=0.75) +
  geom_point(data=slice(P_df_2, ix_2), aes(x=x, y=y), color='red', size=0.75) +
  annotate('text', x=P_df_2$x[ix_2]+0.07, y=P_df_2$y[ix_2]-300000,
           label=TeX(sprintf('$\\omega = %.4f$', P_df_1$x[ix_2])), size=3) +
  geom_line(size=0.25) +
  scale_y_continuous(labels = function(x) format(x, scientific = TRUE)) +
  th + labs(x='', y='Periodogram', title='Scaled Periodogram for June 3rd-7th, 2019')

g3 <- ggplot(P_df_3, aes(x=x, y=y)) +
  geom_point(shape=1, size=0.75) +
  geom_point(data=slice(P_df_3, ix_3), aes(x=x, y=y), color='red', size=0.75) +
  annotate('text', x=P_df_3$x[ix_3]+0.07, y=P_df_3$y[ix_3]-300000,
           label=TeX(sprintf('$\\omega = %.4f$', P_df_3$x[ix_3])), size=3) +
  geom_line(size=0.25) +
  scale_y_continuous(labels = function(x) format(x, scientific = TRUE)) +
  labs(x='Frequency', y='Periodogram', title='Scaled Periodogram for June 5th-7th, 2019') +
  th

floor(1/P_df_1$x[which.max(P_df_1$y)])
floor(1/P_df_2$x[which.max(P_df_2$y)])
floor(1/P_df_3$x[which.max(P_df_3$y)])

g <- arrangeGrob(g1, g2, g3)
#ggsave('../plots/pgrams.png', g, bg='transparent')
```

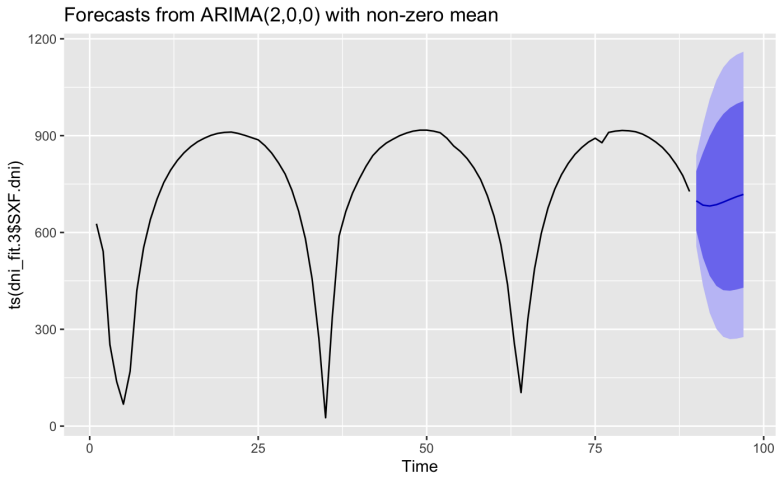
A.3 Frequency-Domain Forecasting

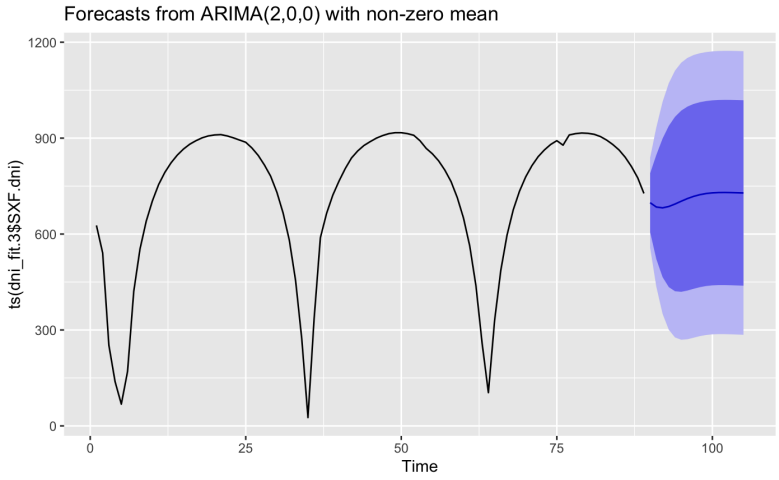
Fit Differenced ARIMA Models and Forecast

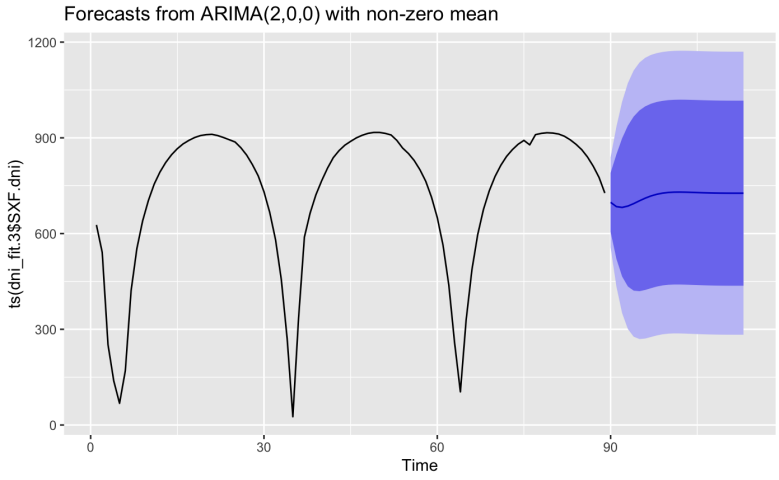
```
# differencing over a lag of 29
dni_diff.7 = diff(dni_fit.7$SXF.dni, 29)
dni_diff.5 = diff(dni_fit.5$SXF.dni, 29)
dni_diff.3 = diff(dni_fit.3$SXF.dni, 29)

# fit ARIMA models
ARIMA.7 = auto.arima(ts(dni_diff.7))
ARIMA.5 = auto.arima(ts(dni_diff.5))
ARIMA.3 = auto.arima(ts(dni_diff.3))

# 4-hr pred
pred.7.4hr = predict(ARIMA.7, n.ahead = 8)
pred.5.4hr = predict(ARIMA.5, n.ahead = 8)
pred.3.4hr = predict(ARIMA.3, n.ahead = 8)

# 8-hr pred
pred.7.8hr = predict(ARIMA.7, n.ahead = 16)
pred.5.8hr = predict(ARIMA.5, n.ahead = 16)
pred.3.8hr = predict(ARIMA.3, n.ahead = 16)

# 12-hr pred
pred.7.12hr = predict(ARIMA.7, n.ahead = 24)
pred.5.12hr = predict(ARIMA.5, n.ahead = 24)
pred.3.12hr = predict(ARIMA.3, n.ahead = 24)

# 4-hour prediction errors

MBE.7.4hr = mean(pred.7.4hr$pred - dni_for.4hr$SXF.dni)
RMSE.7.4hr = sqrt(mean((pred.7.4hr$pred - dni_for.4hr$SXF.dni)^2))
MAE.7.4hr = mean(abs(pred.7.4hr$pred - dni_for.4hr$SXF.dni))

MBE.5.4hr = mean(pred.5.4hr$pred - dni_for.4hr$SXF.dni)
RMSE.5.4hr = sqrt(mean((pred.5.4hr$pred - dni_for.4hr$SXF.dni)^2))
MAE.5.4hr = mean(abs(pred.5.4hr$pred - dni_for.4hr$SXF.dni))

MBE.3.4hr = mean(pred.3.4hr$pred - dni_for.4hr$SXF.dni)
RMSE.3.4hr = sqrt(mean((pred.3.4hr$pred - dni_for.4hr$SXF.dni)^2))
MAE.3.4hr = mean(abs(pred.3.4hr$pred - dni_for.4hr$SXF.dni))

# 8-hour prediction errors

MBE.7.8hr = mean(pred.7.8hr$pred - dni_for.8hr$SXF.dni)
RMSE.7.8hr = sqrt(mean((pred.7.8hr$pred - dni_for.8hr$SXF.dni)^2))
MAE.7.8hr = mean(abs(pred.7.8hr$pred - dni_for.8hr$SXF.dni))

MBE.5.8hr = mean(pred.5.8hr$pred - dni_for.8hr$SXF.dni)
RMSE.5.8hr = sqrt(mean((pred.5.8hr$pred - dni_for.8hr$SXF.dni)^2))
MAE.5.8hr = mean(abs(pred.5.8hr$pred - dni_for.8hr$SXF.dni))

MBE.3.8hr = mean(pred.3.8hr$pred - dni_for.8hr$SXF.dni)
RMSE.3.8hr = sqrt(mean((pred.3.8hr$pred - dni_for.8hr$SXF.dni)^2))
MAE.3.8hr = mean(abs(pred.3.8hr$pred - dni_for.8hr$SXF.dni))

# 12-hour prediction errors

MBE.7.12hr = mean(pred.7.12hr$pred - dni_for.12hr$SXF.dni)
```

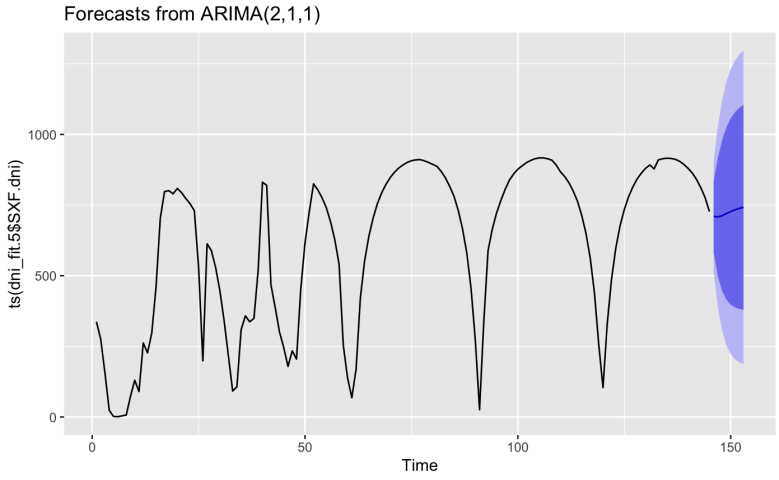
B Additional Figures

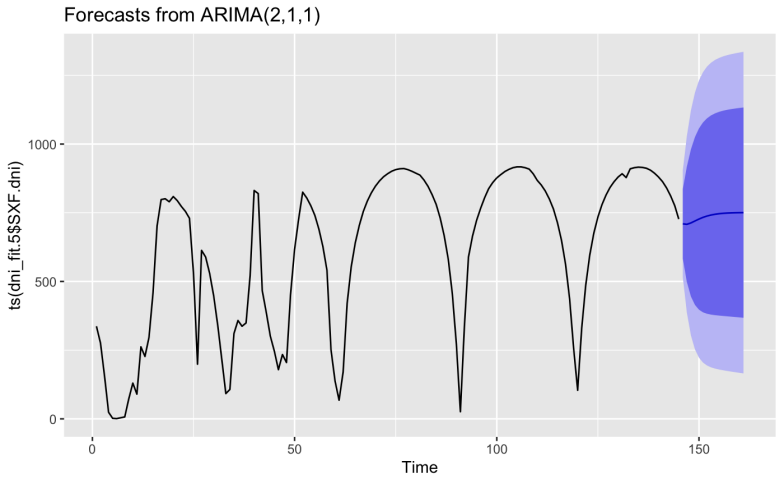
B.1 Original Forecast Plots

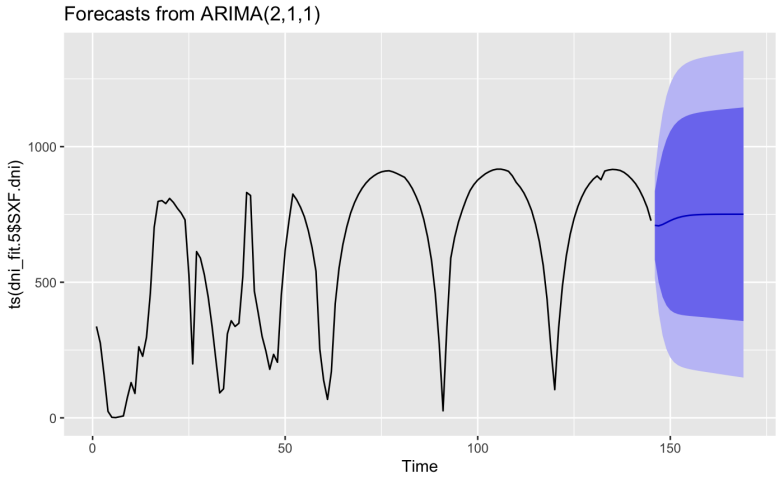


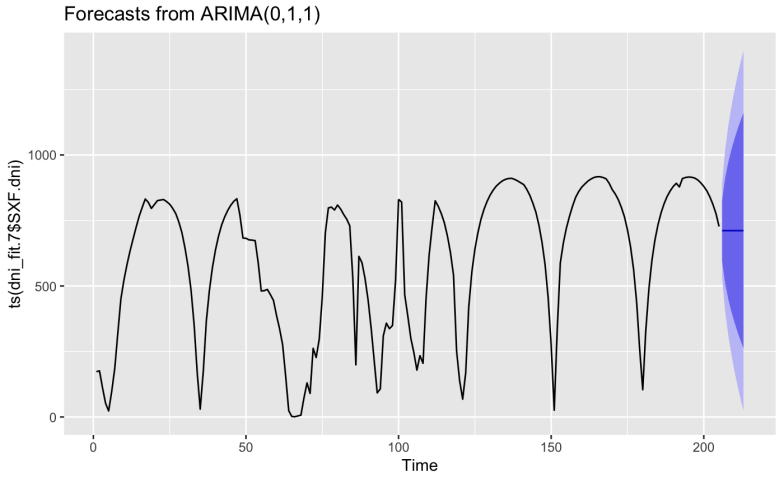


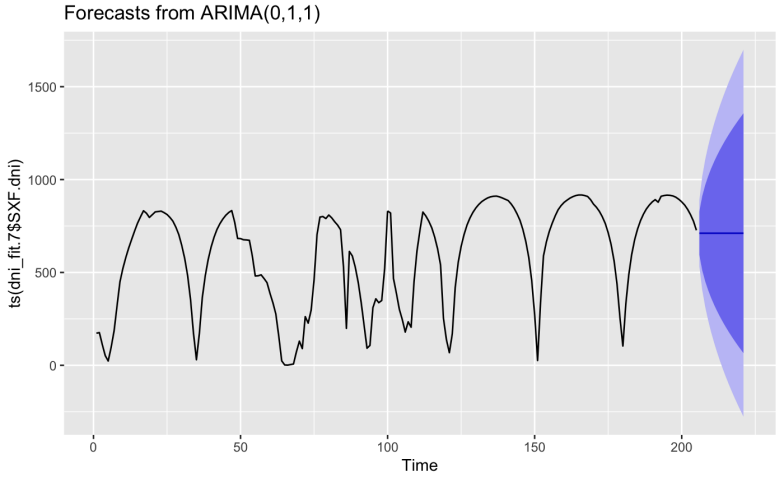


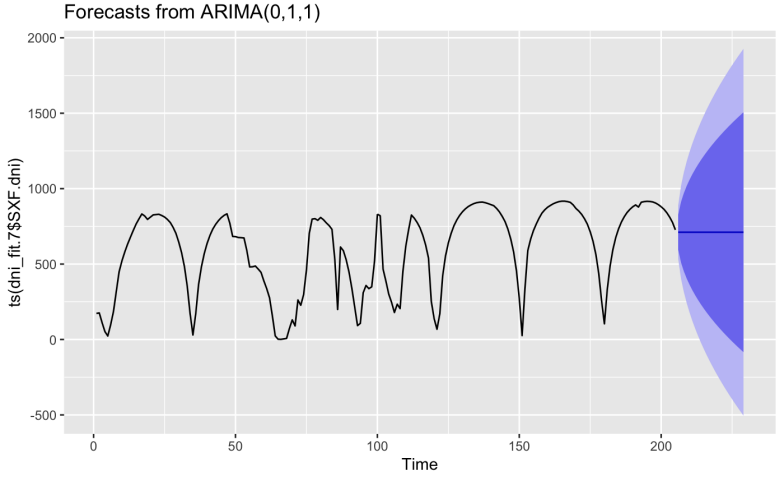

Forecasts from ARIMA(2,1,1)
ts(dni_fit.5$SXF.dni)
1000
500
0
0
50
100
150
Time


Forecasts from ARIMA(2,1,1)
ts(dni_fit.5$SXF.dni)
1000
500
0
0
50
100
150
Time

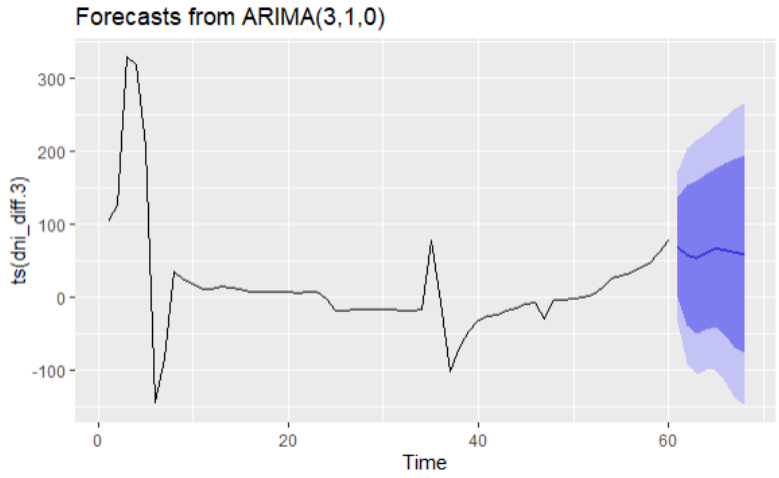

Forecasts from ARIMA(2,1,1)
ts(dni_fit.5$SXF.dni)
1000
500
0
0
50
100
150
Time

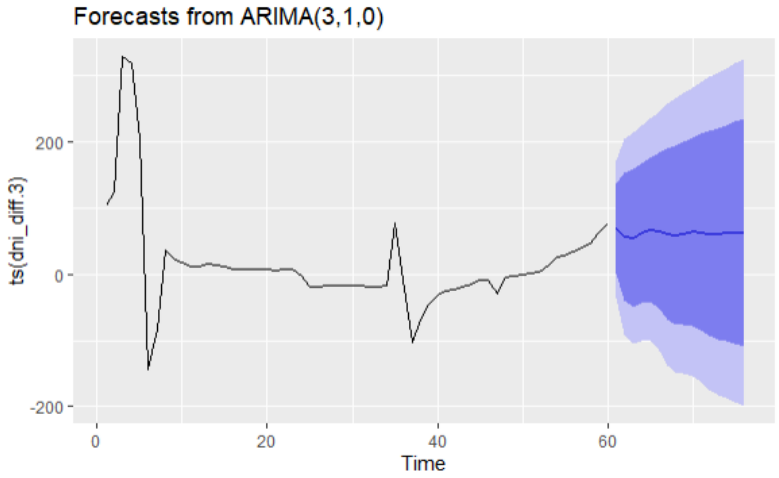

Forecasts from ARIMA(0,1,1)
ts(dni_fit.7$SXF.dni)
1000
500
0
0
50
100
150
200
Time

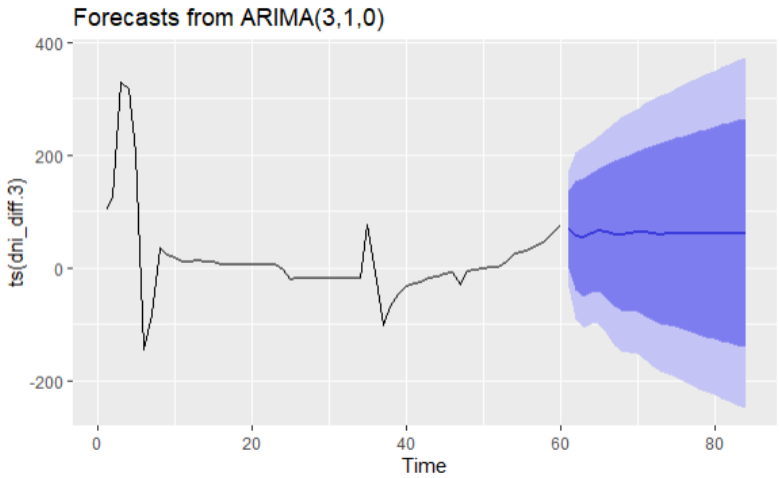

Forecasts from ARIMA(0,1,1)
ts(dni_fit.7$SXF.dni)
1500
1000
500
0
0
50
100
150
200
Time

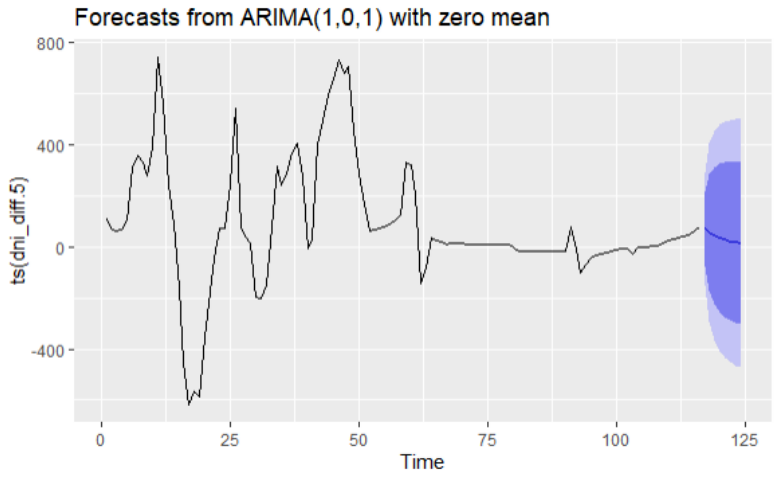

Forecasts from ARIMA(0,1,1)
ts(dni_fit.7$SXF.dni)
2000
1500
1000
500
0
-500
0
50
100
150
200
Time

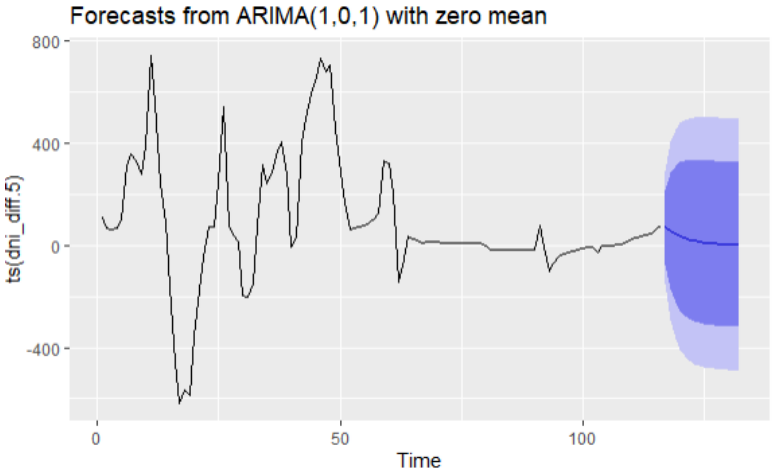
B.2 Differenced Forecast Plots

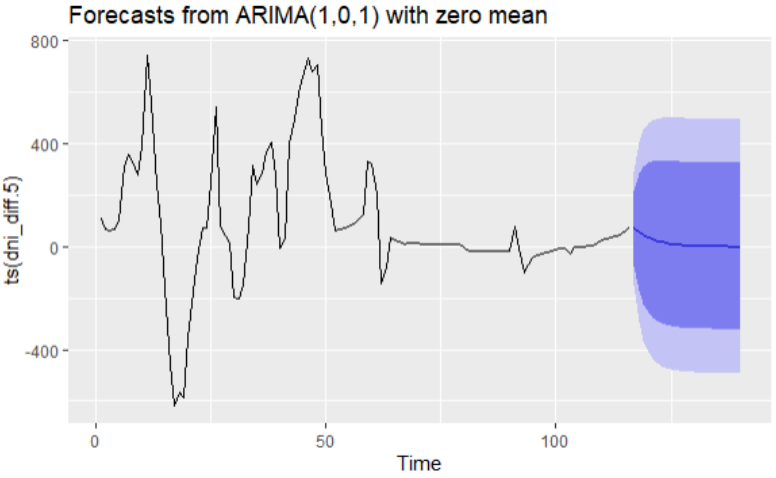


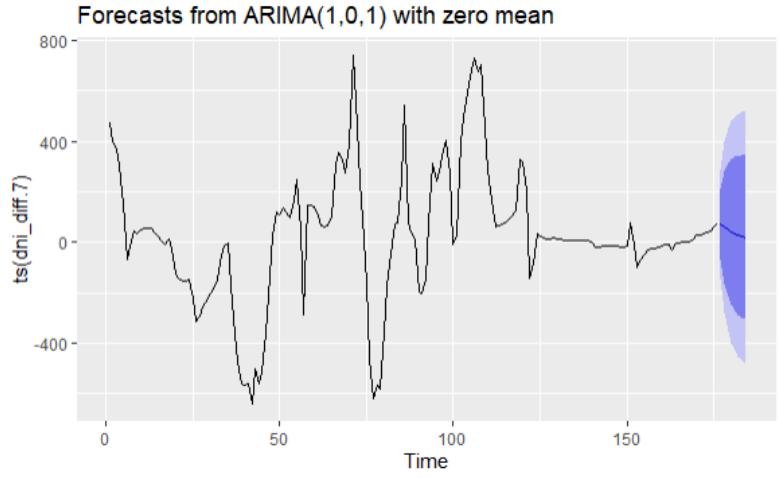


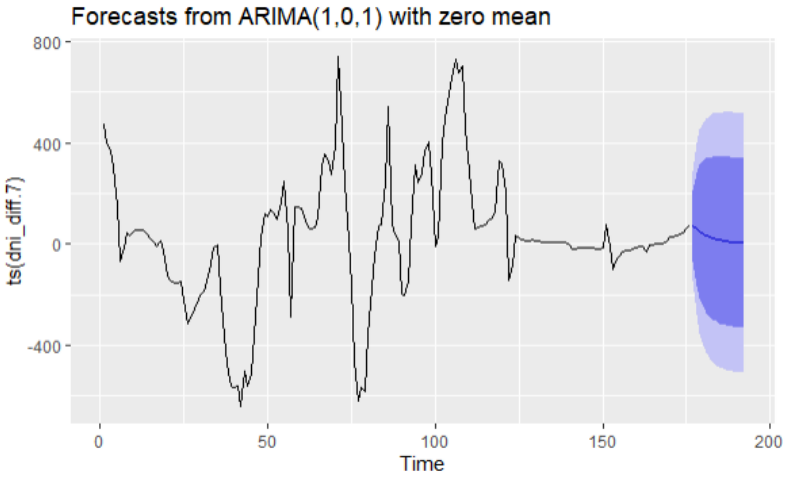


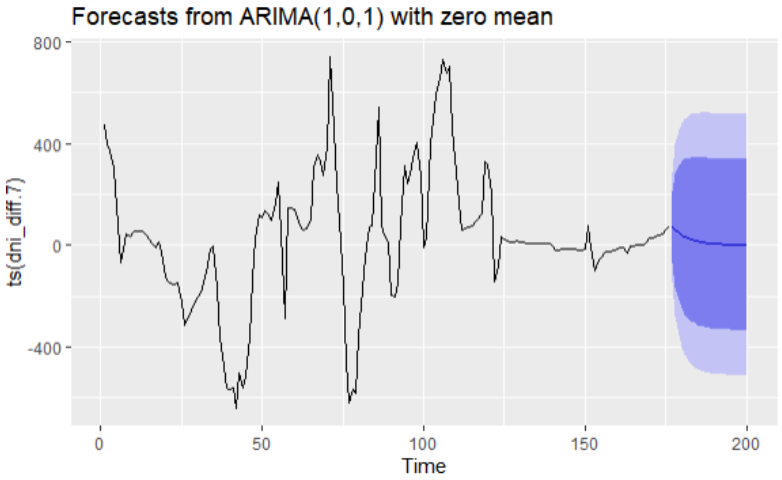

Forecasts from ARIMA(1,0,1) with zero mean
800
400
0
-400
ts(dni_diff.5)
0
25
50
75
100
125
Time


Forecasts from ARIMA(1,0,1) with zero mean
800
400
0
-400
ts(dni_diff.5)
0
50
100
Time


Forecasts from ARIMA(1,0,1) with zero mean
800
400
0
-400
ts(dni_diff.5)
0
50
100
Time


Forecasts from ARIMA(1,0,1) with zero mean
800
400
0
-400
ts(dni_diff.7)
0
50
100
150
Time


Forecasts from ARIMA(1,0,1) with zero mean
800
400
0
-400
ts(dni_diff.7)
0
50
100
150
200
Time


Forecasts from ARIMA(1,0,1) with zero mean
800
400
0
-400
ts(dni_diff.7)
0
50
100
150
200
Time

References

[1] Gordon Reikard and Clifford Hansen. "Forecasting solar irradiance at short horizons: Frequency and time domain models". In: *Renewable Energy* 135 (May 2019), pp. 1270–1290. DOI: 10.1016/j.renene.2018.08.081. URL: https://doi.org/10.1016/j.renene.2018.08.081.

[2] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications*. Springer International Publishing, 2017. DOI: 10.1007/978-3-319-52452-8. URL: https://doi.org/10.1007/978-3-319-52452-8.

[3] Qian Zhang, Yuan Ma, Guoli Li, Jinhui Ma, and Jinjin Ding. "Short-Term Load Forecasting Based on Frequency Domain Decomposition and Deep Learning". In: *Mathematical Problems in Engineering* 2020 (Feb. 2020), pp. 1–9. DOI: 10.1155/2020/7240320. URL: https://doi.org/10.1155/2020/7240320.