3 - Modelos no paramétricos

Definiremos las siguientes funciones para $u(t), y(t)$:

- Autocorrelación de $u(t)$:
$$\hat{R}_u^N(\tau) = \frac{1}{N}\sum_{t=\tau}^{N} u(t)u(t-\tau)$$
- Espectro de potencia de $u(t)$:
$$\Phi_u^N(\omega) = \sum_{\tau=-\gamma}^{\gamma} W_\gamma(\tau)\hat{R}_u^N(\tau)e^{-i\omega\tau}$$
- Correlación cruzada entre $y(t)$ y $u(t)$,
$$\hat{R}_{yu}^N(\tau) = \frac{1}{N}\sum_{t=\tau}^{N} y(t)u(t-\tau)$$
- Espectro de potencia de cruazado entre $y(t)$ y $u(t)$:
$$\Phi_{yu}^N(\omega) = \sum_{\tau=-\gamma}^{\gamma} W_\gamma(\tau)\hat{R}_{yu}^N(\tau)e^{-i\omega\tau}$$

Cada señalar que estas esta definiciones se consideran con media cero, es decir que previamente se debe computar:

$$u(t) = u_{medido}(t) - \mu_u$$
$$y(t) = y_{medido}(t) - \mu_y$$

siendo μ_u, μ_y la media de cada función.

In [1]:
```python
# Importamos las librerias necesarias para trabajar
from statsmodels.regression.linear_model import yule_walker
from statsmodels.graphics.tsaplots import plot_acf
from sklearn.metrics import mean_squared_error
from numpy.fft import fft, fftfreq, fftshift
from datetime import time
from matplotlib import rc
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import mysql.connector
import seaborn as sns
import pandas as pd
import numpy as np
import datetime
import copy
import sklearn

# Seteamos el estilo de los graficos
sns.set(style="whitegrid")

# Configuramos los graficos con latex
plt.rc('text', usetex=True)
```

Abrimos la primera base de datos (proveniente del sensor continuo de glucosa)

In [2]:
```
# Abrimos la base de datos
mydb = mysql.connector.connect(
    host='localhost',
    user='root',
    password='7461143',
    database='datos_ordenados'
)

# Extraemos la informacion en un dataframe
df = pd.read_sql("SELECT * FROM cgm_ordenados", mydb)   # Cargamos todos los d
atos
#df.drop('id', axis=1, inplace=True)                        # Eliminamos el indice
df.set_index('datetime', inplace=True)                     # Definimos datetime com
o indice
df.sort_index(inplace=True)                                # Ordenamos en base a da
tetime
df.index.freq = pd.infer_freq(df.index)
# Mostramos los resultados
print('Tamano de la tabla: {} filas y {} columnas'.format(df.shape[0], df.shap
e[1]))
print('Tiempo del estudio:')
print(' - Inicio  : {}'.format(str(df.index[0])))
print(' - Final   : {}'.format(str(df.index[-1])))
print(' - Duración: {}'.format(str(df.index[-1] - df.index[0])))
df.head(3)
```

```
Tamano de la tabla: 1728 filas y 6 columnas
Tiempo del estudio:
 - Inicio  : 2020-01-24 17:00:00
 - Final   : 2020-01-30 16:55:00
 - Duración: 5 days 23:55:00
```

Out[2]:

datetime	sensor_glucose	sensor_calibration_bg	meal	basal_insulin	bolus_insulin	exercise
2020-01-24 17:00:00	NaN	125.0	NaN	NaN	NaN	NaN
2020-01-24 17:05:00	126.0	NaN	NaN	NaN	NaN	NaN
2020-01-24 17:10:00	128.0	NaN	NaN	NaN	NaN	NaN

Extraemos las señales señales de interés. Estas son:

- `sensor_glucose`
- `meal`
- `basal_insulin`
- `bolus_insulin`

In [132]:
```
# Obtenemos los datos
y = copy.copy(df['sensor_glucose'])
u_meal = copy.copy(df['meal'])
u_basal_insulin = copy.copy(df['basal_insulin'])
u_bolus_insulin = copy.copy(df['bolus_insulin'])
```

Adicionalmente, creamos una función que calcula el espectro

In [151]:
```
def phi_X(R_X, gamma, Ts=5*60):
    arg_max = R_X.argmax()
    R_X_wind = R_X[arg_max - gamma: arg_max + gamma + 1]
    wind = np.hanning(len(R_X_wind))
    phi_X = fft(R_X_wind * wind)
    freq = fftfreq(len(phi_X), Ts)
    phi_X = pd.Series(phi_X, index=freq)
    phi_X = phi_X[freq > 0]
    return phi_X
```

3.1 Señal de salida - Sensor de glucosa

La señal de salida es la que proviene del sensor continuo de glucosa en la variable `y` . Esta señal tiene unas pequeñas pérdidas de información que serán interpoladas linealmente.

In [133]:
```
# Realizamos la interpolacion
y.interpolate(inplace=True, limit_direction='both')
# Eliminamos los valores vacios
y.head(5)
```

Out[133]:
```
datetime
2020-01-24 17:00:00    126.0
2020-01-24 17:05:00    126.0
2020-01-24 17:10:00    128.0
2020-01-24 17:15:00    146.0
2020-01-24 17:20:00    158.0
Freq: 5T, Name: sensor_glucose, dtype: float64
```

- Calculamos la autocorrelación $\hat{R}_y^N(\tau)$. Recordar que la frecuencia de la señal es de 5 minutos. Se utilizará la función de numpy `correlate` , donde previamente se verificó que realiza el mismo que la ecuación planteada

In [150]:
```
# Computo manual
y_1 = y - y.mean()
N = len(y_1)

# Computo con la funcion correlacion
R_y1 = np.correlate(y_1, y_1, mode='full') / N
```

- Caculamos el periodograma $|Y_N(\omega)|^2$

In [137]:
```
freq = fftfreq(N, 5*60)
Y = fft(y_1, norm='ortho')
Y_N = abs(Y) ** 2
Y_N = pd.Series(Y_N, index=freq)
Y_N = Y_N[freq > 0]
```

- Calculamos el espectro Φ_y^N para distintos γ:

In [152]:
```
N = len(R_y1)
# Gamma = N/2
gamma = round(N / 2) - 1
phi_Y1 = phi_X(R_y1, gamma)

# Gamma = N/3
gamma = round(N / 3)
phi_Y2 = phi_X(R_y1, gamma)

# Gamma = N/4
gamma = round(N / 4)
phi_Y3 = phi_X(R_y1, gamma)

# Gamma = N/5
gamma = round(N / 5)
phi_Y4 = phi_X(R_y1, gamma)

# Gamma = N/6
gamma = round(N / 6)
phi_Y5 = phi_X(R_y1, gamma)

# Gamma = N/10
gamma = round(N / 10)
phi_Y6 = phi_X(R_y1, gamma)
```

Graficos

Gráfico en el tiempo de $y(t)$

In [168]:
```
# Creamos la figura
fig, ax = plt.subplots()

# Graficamos la senal
ax.plot(y, color='#600000')

# Seteamos los parametros
ax.set_ylim([0, 450])
ax.set_yticks(np.arange(0, 450, 50))
ax.set_ylabel('Glucose [mg/dL]')
ax.set_xlabel('Dates')
ax.grid(True)


date_form = mdates.DateFormatter('%d-%m-%Y')
ax.xaxis.set_major_formatter(date_form)

y_size = 4.2
x_size = 3 * y_size
fig.set_size_inches(x_size, y_size)
plt.tight_layout()
```

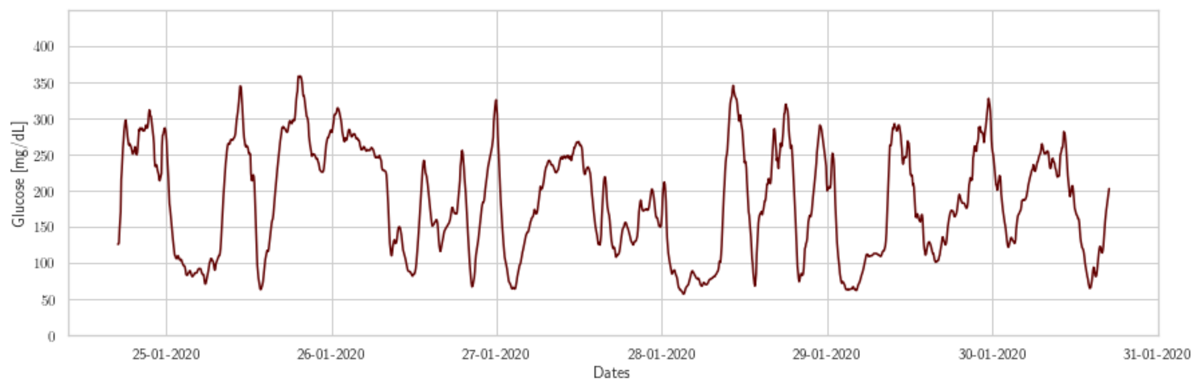


Gráfico de correlación $R_y(\tau)$

In [239]:
```
# Creamos la figura y el axis
fig, ax = plt.subplots()

# Generamos las variables a graficar
tau_0 = R_u_meal.argmax()
tau = np.array(list(range(len(R_u_meal)))) - tau_0
tau_horas = tau * 5 / 60

# Realizamos el grafico
ax.plot(tau_horas[tau_0:], R_y1[tau_0:], color='#600000')

# Configuramos los parametros
ax.set_xticks(np.arange(-144, 145, 6))
ax.set_ylabel(r'Glucose$^2$ [mg$^2$/dL$^2$]')
ax.set_xlabel(r'Lag $\tau$ [Hour]')
ax.set_xlim([-3, 147])
ax.grid(True)


y_size = 4.2
x_size = 3 * y_size
fig.set_size_inches(x_size, y_size)
plt.tight_layout()
```

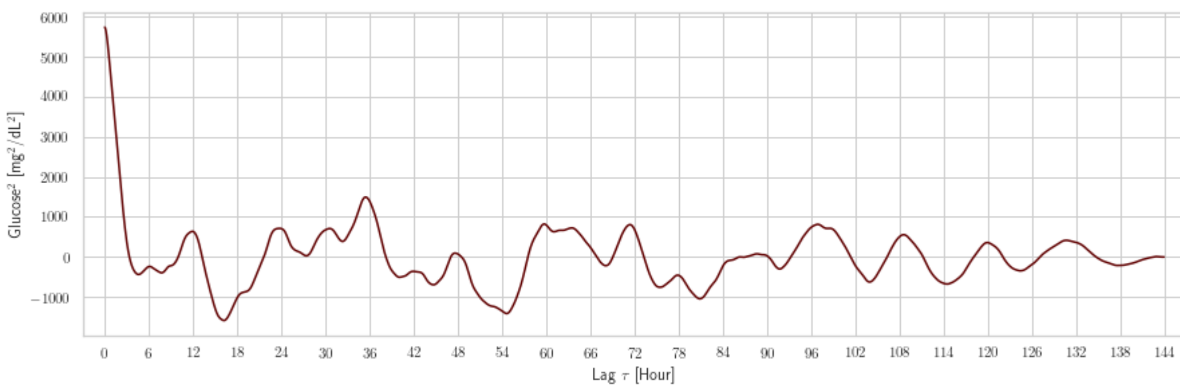
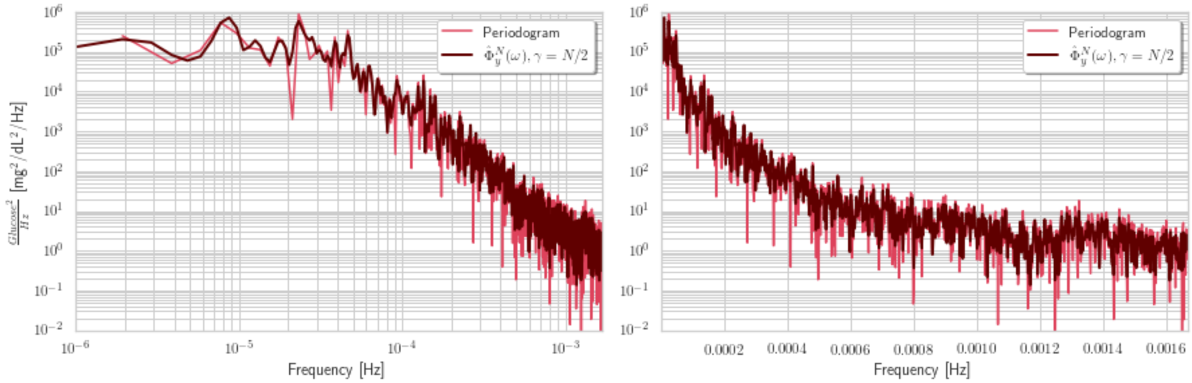
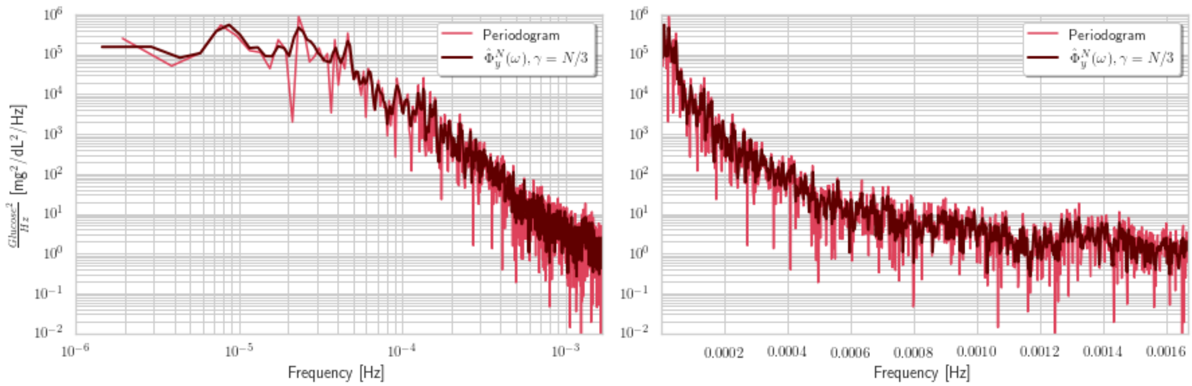
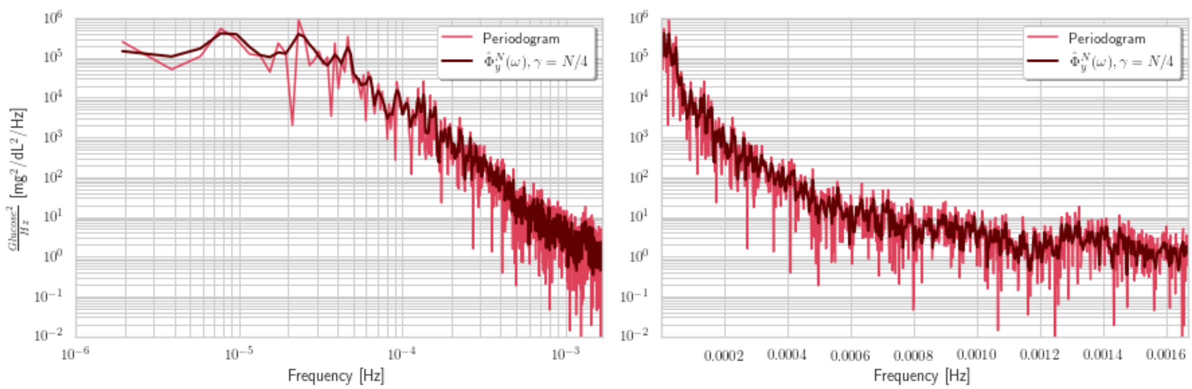


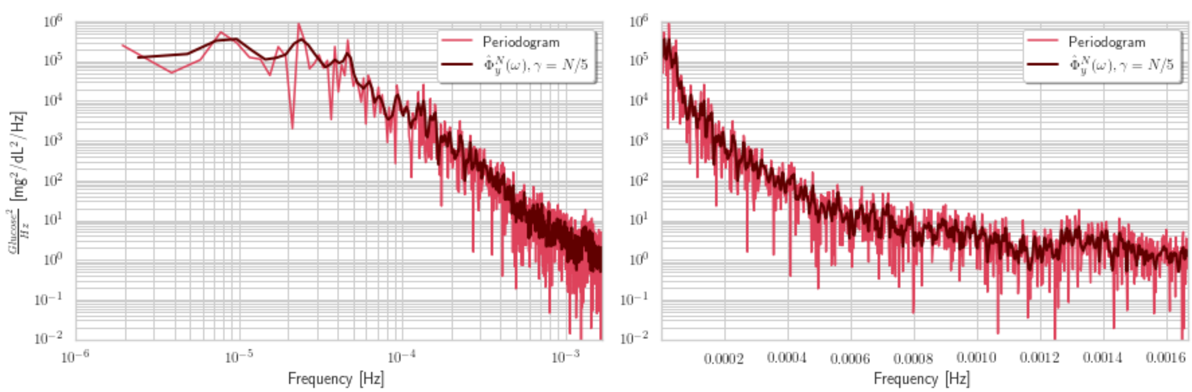
Gráfico de periodogramaba y estimación del espectro $\hat{\Phi}_y^N(\omega)$ para distintos γ

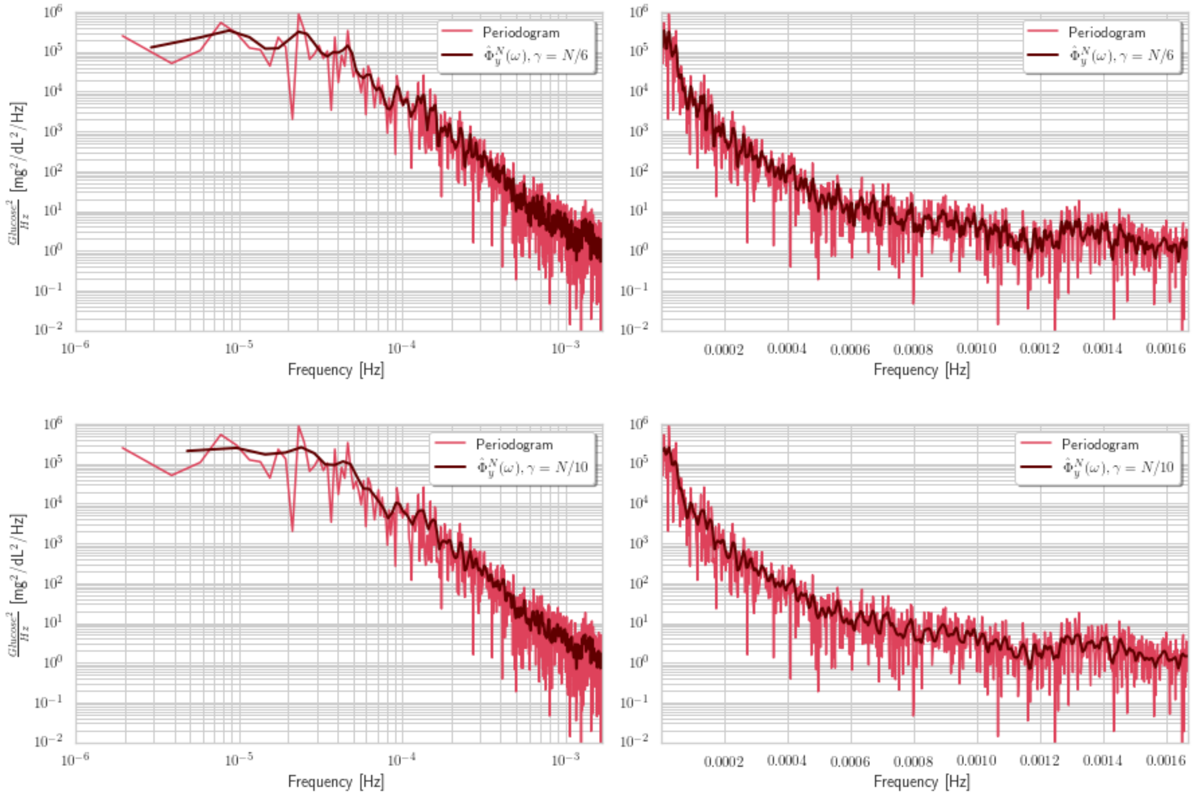
In [270]:
```
phi_list = [phi_Y1, phi_Y2, phi_Y3, phi_Y4, phi_Y5, phi_Y6]
gamma_list = ['N/2', 'N/3', 'N/4', 'N/5', 'N/6', 'N/10']

for i in range(len(phi_list)):
    phi_buff = phi_list[i]
    gamma_buff = gamma_list[i]

    # Creamos la figura y el axis
    fig, (ax1, ax2) = plt.subplots(1, 2)

    # Realizamos el grafico
    label_name = r'$\hat{\Phi}^N_y(\omega), \gamma = ' + gamma_buff + '$'
    ax1.loglog(Y_N, color='#DE425B', label=r'Periodogram')
    ax1.loglog(abs(phi_buff), color='#600000', lineWidth=2, label=label_name)

    ax2.semilogy(Y_N, color='#DE425B', label=r'Periodogram')
    ax2.semilogy(abs(phi_buff), color='#600000', lineWidth=2, label=label_name
)

    # Configuramos los parametros
    ax1.grid(True, which='both')
    ax1.set_ylim([10 ** (-2), 10 ** 6])
    ax1.set_xlim([10 ** (-6), max(Y_N.index)])
    ax1.legend(fancybox=True, shadow=True)

    ax2.grid(True, which='both')
    ax2.set_ylim([10 ** (-2), 10 ** 6])
    ax2.set_xlim([10 ** (-6), max(Y_N.index)])
    ax2.legend(fancybox=True, shadow=True)

    ax1.set_ylabel(r'$\frac{Glucose^2}{Hz}$ [mg$^2$/dL$^2$/Hz]')
    ax1.set_xlabel(r'Frequency [Hz]')
    ax2.set_xlabel(r'Frequency [Hz]')

    x_size = 3 * 4.2
    y_size = 1 * x_size / 3
    fig.set_size_inches(x_size, y_size)
    plt.tight_layout()
```


$\frac{Glucose^2}{Hz}$ [mg^2/dL^2/Hz]
Periodogram
$\hat{\Phi}_y^N(\omega), \gamma = N/2$
Frequency [Hz]
Frequency [Hz]


$\frac{Glucose^2}{Hz}$ [mg^2/dL^2/Hz]
Periodogram
$\hat{\Phi}_y^N(\omega), \gamma = N/3$
Frequency [Hz]
Frequency [Hz]


$\frac{Glucose^2}{Hz}$ [mg^2/dL^2/Hz]
Periodogram
$\hat{\Phi}_y^N(\omega), \gamma = N/4$
Frequency [Hz]
Frequency [Hz]


$\frac{Glucose^2}{Hz}$ [mg^2/dL^2/Hz]
Periodogram
$\hat{\Phi}_y^N(\omega), \gamma = N/5$
Frequency [Hz]
Frequency [Hz]



Calculamos el periodo de las componentes que cuentan con mayor potencia

In [214]:
```
N_indices = 10
index_ordenado_1 = abs(phi_Y1).sort_values(ascending=False).index[:N_indices]
index_ordenado_2 = abs(phi_Y2).sort_values(ascending=False).index[:N_indices]
index_ordenado_3 = abs(phi_Y3).sort_values(ascending=False).index[:N_indices]
index_ordenado_4 = abs(phi_Y4).sort_values(ascending=False).index[:N_indices]
index_ordenado_5 = abs(phi_Y4).sort_values(ascending=False).index[:N_indices]
index_ordenado_6 = abs(phi_Y4).sort_values(ascending=False).index[:N_indices]
frame = {'N/2':index_ordenado_1,
         'N/3':index_ordenado_2,
         'N/4':index_ordenado_3,
         'N/5':index_ordenado_4,
         'N/6':index_ordenado_5,
         'N/10':index_ordenado_6,
        }

max_index = pd.DataFrame(frame)
max_index.set_index(pd.Index(list(range(1, N_indices + 1))), inplace=True)
max_index = max_index.applymap(lambda x: 1 / x / 60 / 60)

max_index
```

Out[214]:

	N/2	N/3	N/4	N/5	N/6	N/10
1	31.990741	32.013889	36.020833	28.812500	28.812500	28.812500
2	11.996528	12.005208	12.006944	11.525000	11.525000	11.525000
3	35.989583	38.416667	28.816667	38.416667	38.416667	38.416667
4	11.516667	11.299020	11.083333	12.805556	12.805556	12.805556
5	28.791667	27.440476	13.098485	10.477273	10.477273	10.477273
6	12.518116	12.805556	10.291667	23.050000	23.050000	23.050000
7	11.073718	10.671296	24.013889	6.065789	6.065789	6.065789
8	5.998264	6.002604	6.003472	9.604167	9.604167	9.604167
9	9.928161	10.109649	48.027778	57.625000	57.625000	57.625000
10	5.875850	5.820707	9.605556	14.406250	14.406250	14.406250

3.2 Comida

In [225]:
```
u_meal = copy.copy(df['meal'])
u_meal = u_meal.replace(np.nan, 0)
u_meal_1 = copy.copy(u_meal)
u_meal = u_meal - u_meal.mean()
```

- Calculo de correlación $\hat{R}^{N}_{u_{meal}}(\tau)$

In [26]:
```
# Computo con la funcion correlacion
R_u_meal = np.correlate(u_meal, u_meal, mode='full') / N
```

- Calculo del periodograma $\left|U_N^{meal}(\omega)\right|^2$

In [218]:
```
N = len(u_meal)
freq = fftfreq(N, 5*60)
U_meal = fft(u_meal, norm='ortho')
U_meal = abs(U_meal) ** 2
U_meal = pd.Series(U_meal, index=freq)
U_meal = U_meal[freq > 0]
```

- Calculamos el espectro $\hat{\Phi}_{u_{meal}}^N$ para distintos γ:

In [219]:
```
N = len(R_u_meal)
# Gamma = N/2
gamma = round(N / 2) - 1
phi_U_meal_1 = phi_X(R_u_meal, gamma)

# Gamma = N/3
gamma = round(N / 3)
phi_U_meal_2 = phi_X(R_u_meal, gamma)

# Gamma = N/4
gamma = round(N / 4)
phi_U_meal_3 = phi_X(R_u_meal, gamma)

# Gamma = N/5
gamma = round(N / 5)
phi_U_meal_4 = phi_X(R_u_meal, gamma)

# Gamma = N/6
gamma = round(N / 6)
phi_U_meal_5 = phi_X(R_u_meal, gamma)

# Gamma = N/10
gamma = round(N / 10)
phi_U_meal_6 = phi_X(R_u_meal, gamma)
```

Gráficos

Gráfico en el tiempo de $u_{meal}(t)$

In [231]:
```
# Creamos la figura
fig, ax = plt.subplots()

# Graficamos la senal
ax.plot(u_meal_1, color='g')

ax.set_ylabel('Carbohydrate [Gram]')
ax.set_xlabel('Dates')
ax.grid(True)

date_form = mdates.DateFormatter('%d-%m-%Y')
ax.xaxis.set_major_formatter(date_form)

y_size = 4.2
x_size = 3 * y_size
fig.set_size_inches(x_size, y_size)
plt.tight_layout()
```

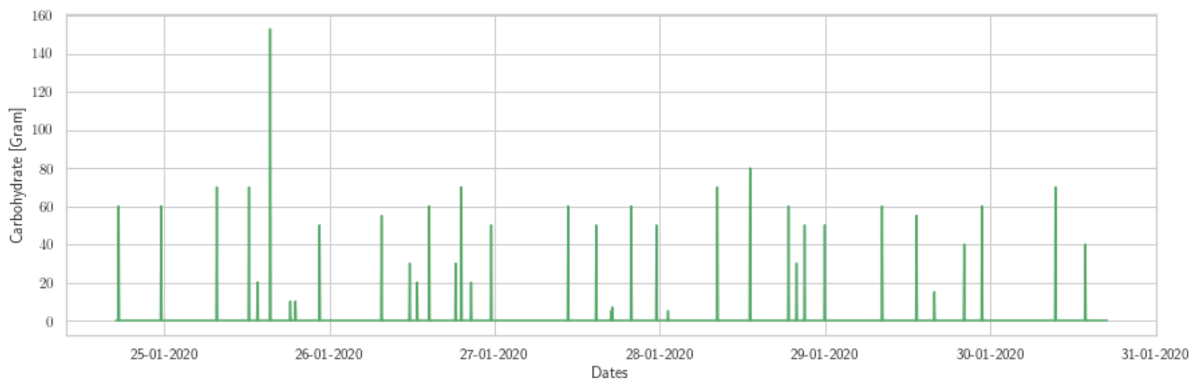


Gráfico de correlación $R_{u_{meal}}(\tau)$

In [242]:
```
# Creamos la figura y el axis
fig, ax = plt.subplots()

# Generamos las variables a graficar
tau_0 = R_u_meal.argmax()
tau = np.array(list(range(len(R_u_meal)))) - tau_0
tau_horas = tau * 5 / 60

# Realizamos el grafico
ax.plot(tau_horas[tau_0:], R_u_meal[tau_0:], color='g')

# Configuramos los parametros
ax.set_xticks(np.arange(-144, 145, 6))
ax.set_ylabel(r'Carbohydrate$^2$ [Gram$^2$]')
ax.set_xlabel(r'Lag $\tau$ [Hour]')
ax.set_xlim([-3, 147])
ax.grid(True)


y_size = 4.2
x_size = 3 * y_size
fig.set_size_inches(x_size, y_size)
plt.tight_layout()
```

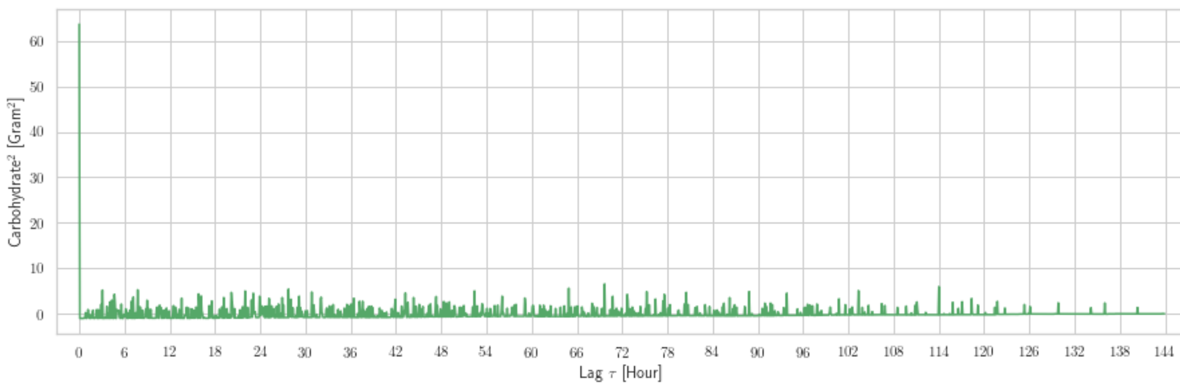
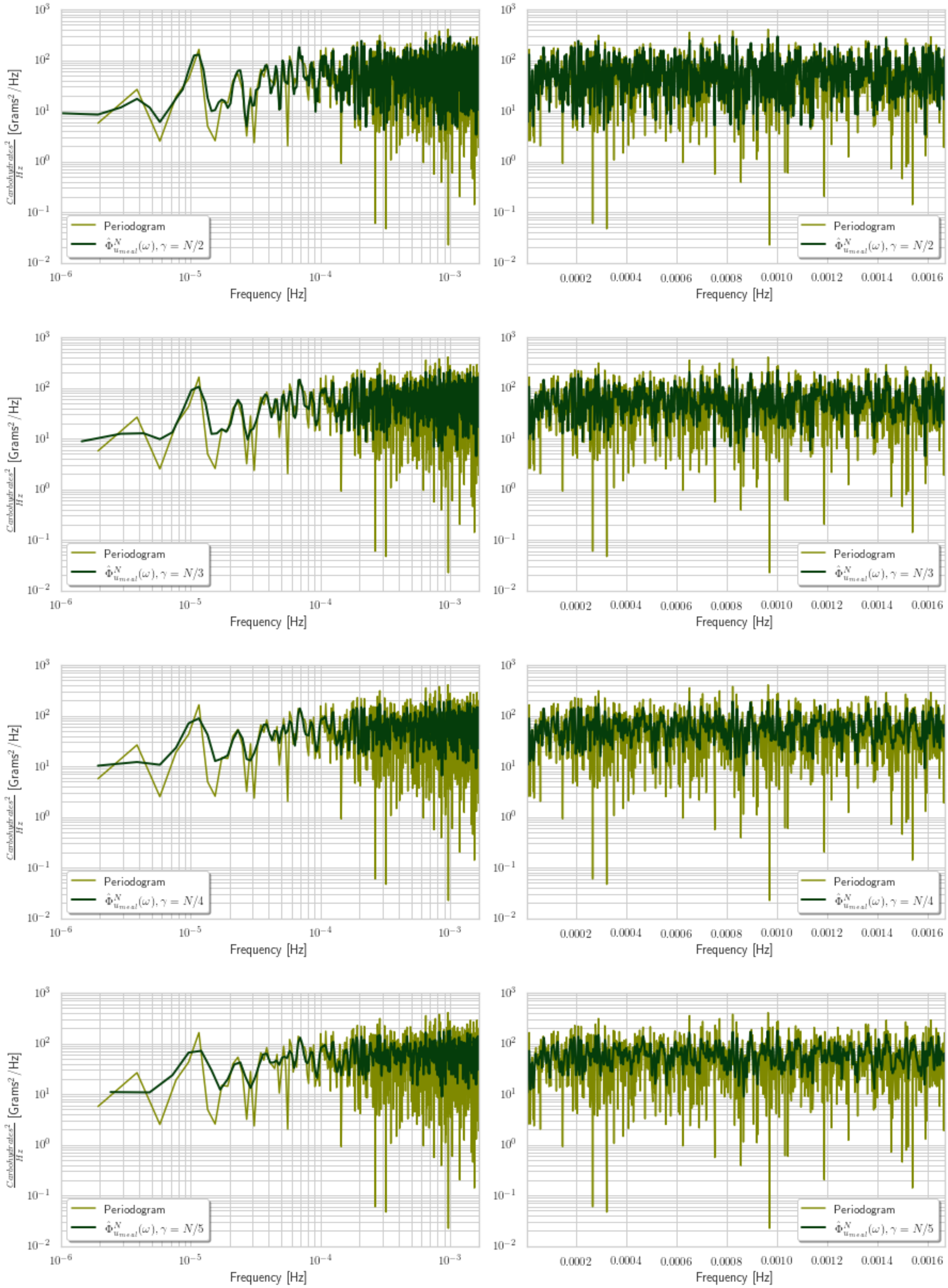
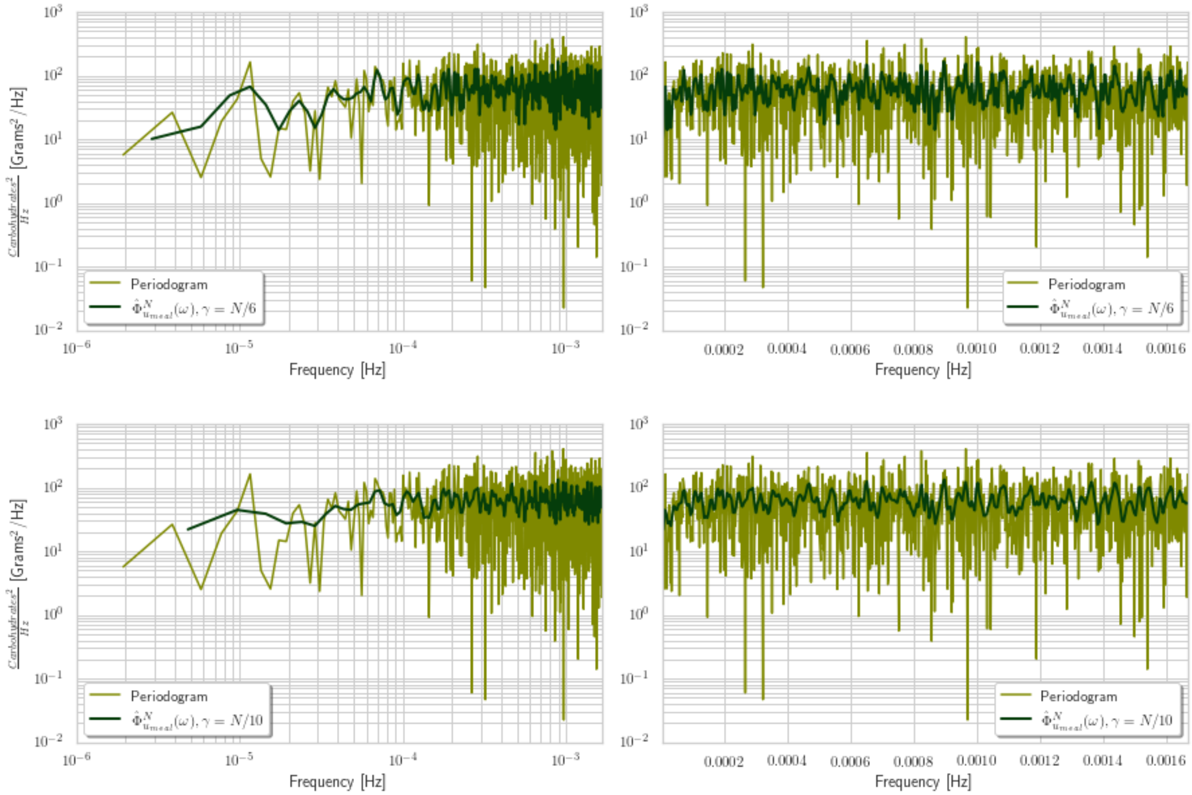


Gráfico de estimación del espectro Φ_u^{meal} y del espectrograma para distintos γ

In [269]:
```
phi_list = [phi_U_meal_1, phi_U_meal_2, phi_U_meal_3, phi_U_meal_4, phi_U_meal
_5, phi_U_meal_6]
gamma_list = ['N/2', 'N/3', 'N/4', 'N/5', 'N/6', 'N/10']

for i in range(len(phi_list)):
    phi_buff = phi_list[i]
    gamma_buff = gamma_list[i]

    # Creamos la figura y el axis
    fig, (ax1, ax2) = plt.subplots(1, 2)

    # Realizamos el grafico
    label_name = r'$\hat{\Phi}^N_{u_{meal}}(\omega), \gamma = ' + gamma_buff +
'$'
    ax1.loglog(U_meal, color='#7f8900', label=r'Periodogram')
    ax1.loglog(abs(phi_buff), color='#053d0b', lineWidth=2, label=label_name)

    ax2.semilogy(U_meal, color='#7f8900', label=r'Periodogram')
    ax2.semilogy(abs(phi_buff), color='#053d0b', lineWidth=2, label=label_name
)

    # Configuramos los parametros
    ax1.grid(True, which='both')
    ax1.set_ylim([10 ** (-2), 10 ** 3])
    ax1.set_xlim([10 ** (-6), max(U_meal.index)])
    ax1.legend(fancybox=True, shadow=True)

    ax2.grid(True, which='both')
    ax2.set_ylim([10 ** (-2), 10 ** 3])
    ax2.set_xlim([10 ** (-6), max(U_meal.index)])
    ax2.legend(fancybox=True, shadow=True)

    ax1.set_ylabel(r'$\frac{Carbohydrates^2}{Hz}$ [Grams$^2$/Hz]')
    ax1.set_xlabel(r'Frequency [Hz]')
    ax2.set_xlabel(r'Frequency [Hz]')

    x_size = 3 * 4.2
    y_size = 1 * x_size / 3
    fig.set_size_inches(x_size, y_size)
    plt.tight_layout()
```


Periodogram
$\hat{\Phi}^N_{u_{meal}}(\omega), \gamma = N/2$
$\hat{\Phi}^N_{u_{meal}}(\omega), \gamma = N/3$
$\hat{\Phi}^N_{u_{meal}}(\omega), \gamma = N/4$
$\hat{\Phi}^N_{u_{meal}}(\omega), \gamma = N/5$
$\frac{Carbohydrates^2}{Hz}$ [Grams^2/Hz]
Frequency [Hz]



3.3 Insulina bolo

In [252]:
```
u_bolus_insulin = copy.copy(df['bolus_insulin'])
u_bolo = u_bolus_insulin.replace(np.nan, 0)
u_bolo_1 = copy.copy(u_bolo)
u_bolo = u_bolo - u_bolo.mean()
```

Calculo de correlación $R_{u_{bolus}}$

In [253]:
```
# Computo con la funcion correlacion
R_u_bolo = np.correlate(u_bolo, u_bolo, mode='full') / N
```

Calculo del periodograma $\left|U_N^{bolus}(\omega)\right|^2$

In [256]:
```
N = len(u_bolo)
freq = fftfreq(N, 5 * 60)
U_bolo = fft(u_bolo, norm='ortho')
U_bolo = abs(U_bolo) ** 2
U_bolo = pd.Series(U_bolo, index=freq)
U_bolo = U_bolo[freq > 0]
```

Calculamos el espectro $\hat{\Phi}^N_{u_{bolus}}$ para distintos γ:

In [262]:
```
N = len(R_u_bolo)
# Gamma = N/2
gamma = round(N / 2) - 1
phi_U_bolo_1 = phi_X(R_u_bolo, gamma)

# Gamma = N/3
gamma = round(N / 3)
phi_U_bolo_2 = phi_X(R_u_bolo, gamma)

# Gamma = N/4
gamma = round(N / 4)
phi_U_bolo_3 = phi_X(R_u_bolo, gamma)

# Gamma = N/5
gamma = round(N / 5)
phi_U_bolo_4 = phi_X(R_u_bolo, gamma)

# Gamma = N/6
gamma = round(N / 6)
phi_U_bolo_5 = phi_X(R_u_bolo, gamma)

# Gamma = N/10
gamma = round(N / 10)
phi_U_bolo_6 = phi_X(R_u_bolo, gamma)
```

Gráficos

Gráfico en el tiempo de $u_{bolus}(t)$

In [260]:
```
# Creamos la figura
fig, ax = plt.subplots()

# Graficamos la senal
ax.plot(u_bolo_1, color='C0')

ax.set_ylabel('Bolus insulin [U]')
ax.set_xlabel('Dates')
ax.grid(True)

date_form = mdates.DateFormatter('%d-%m-%Y')
ax.xaxis.set_major_formatter(date_form)

y_size = 4.2
x_size = 3 * y_size
fig.set_size_inches(x_size, y_size)
plt.tight_layout()
```

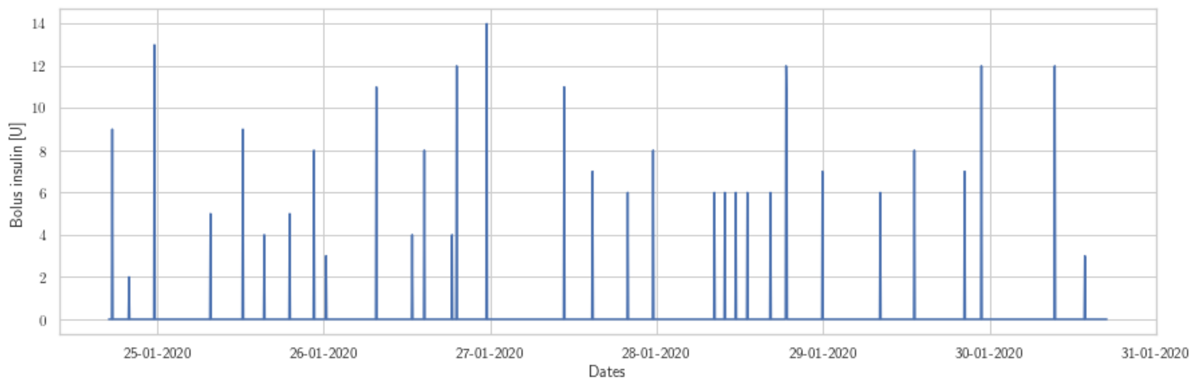


Gráfico de correlación $R_{u_{bolus}}(\tau)$

In [261]:
```
# Creamos la figura y el axis
fig, ax = plt.subplots()

# Generamos las variables a graficar
tau_0 = R_u_bolo.argmax()
tau = np.array(list(range(len(R_u_bolo)))) - tau_0
tau_horas = tau * 5 / 60

# Realizamos el grafico
ax.plot(tau_horas[tau_0:], R_u_bolo[tau_0:], color='C0')

# Configuramos los parametros
ax.set_xticks(np.arange(-144, 145, 6))
ax.set_ylabel(r'Bolus Insulin$^2$ [U$^2$]')
ax.set_xlabel(r'Lag $\tau$ [Hour]')
ax.set_xlim([-3, 147])
ax.grid(True)


y_size = 4.2
x_size = 3 * y_size
fig.set_size_inches(x_size, y_size)
plt.tight_layout()
```

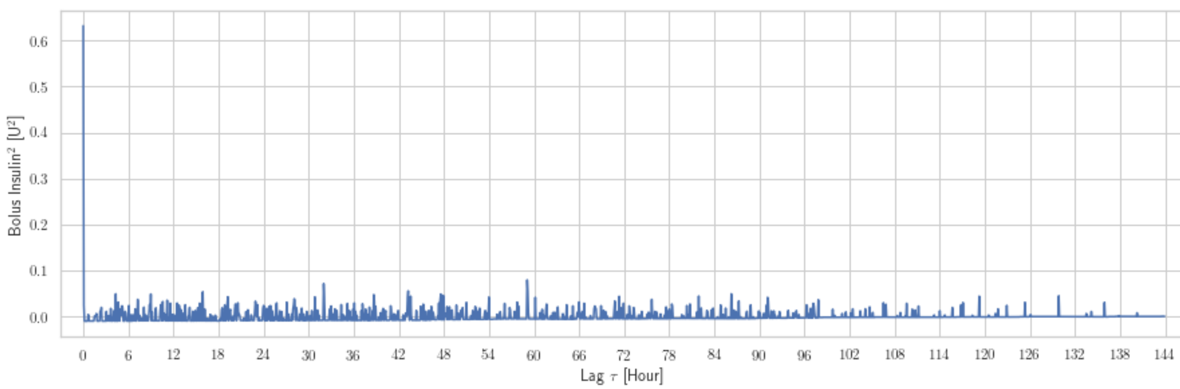
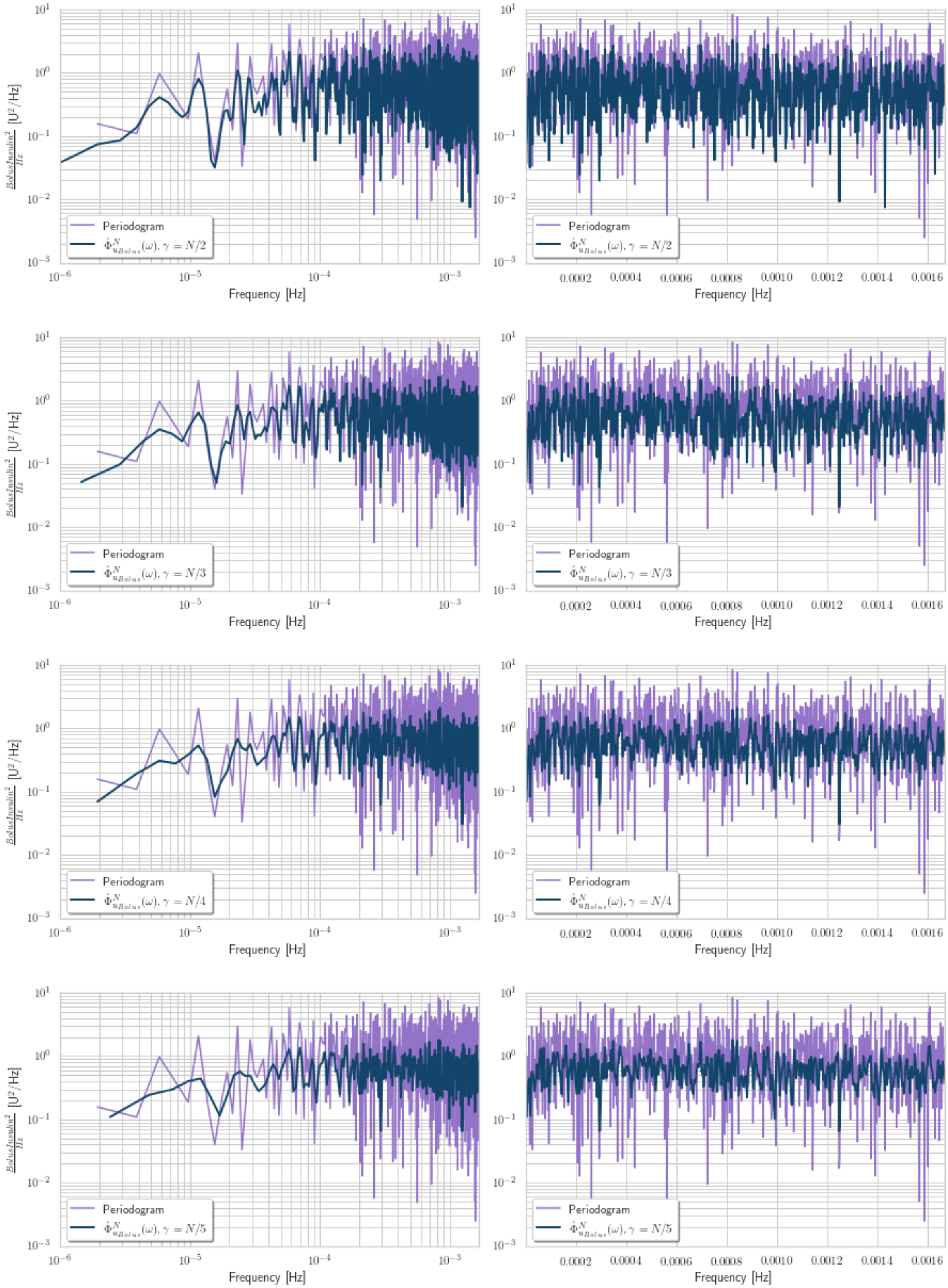
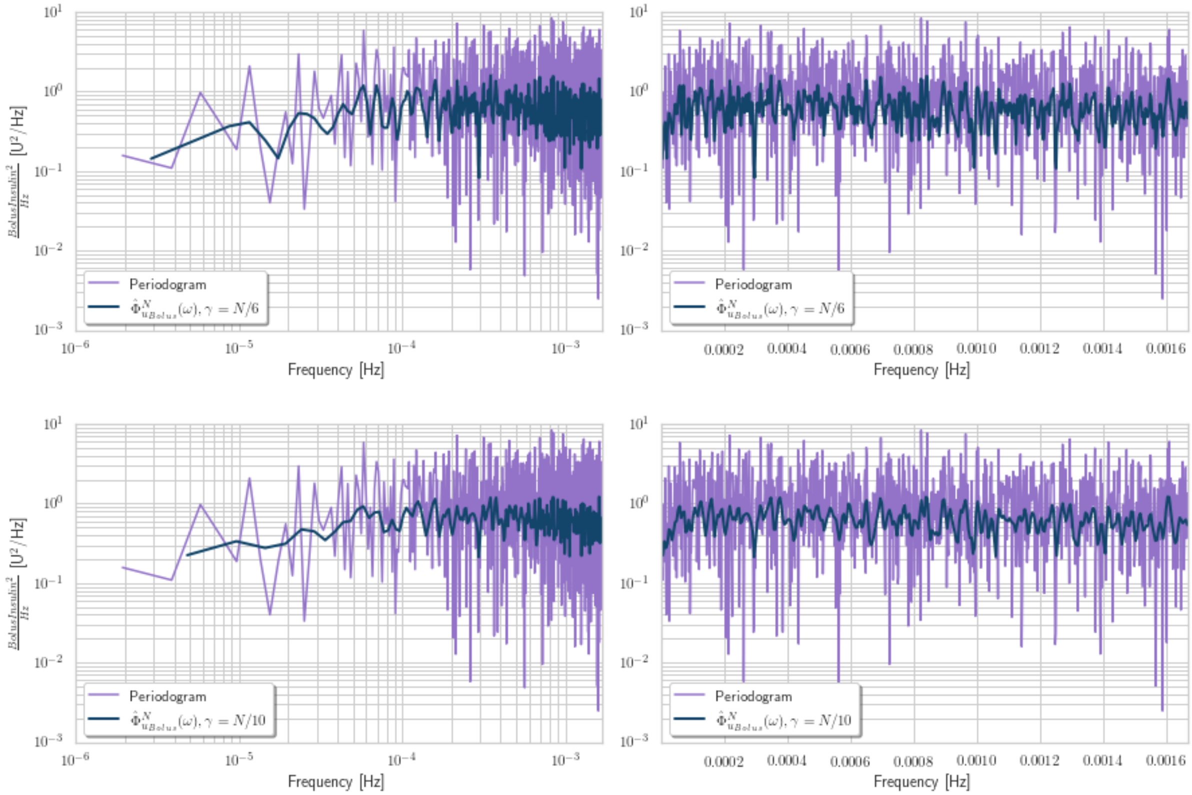


Gráfico de estimación del espectro $\hat{\Phi}^{N}_{u_{bolus}}(\omega)$ y del espectrograma para distintos γ

In [268]:
```
phi_list = [phi_U_bolo_1, phi_U_bolo_2, phi_U_bolo_3, phi_U_bolo_4, phi_U_bolo
_5, phi_U_bolo_6]
gamma_list = ['N/2', 'N/3', 'N/4', 'N/5', 'N/6', 'N/10']

for i in range(len(phi_list)):
    phi_buff = phi_list[i]
    gamma_buff = gamma_list[i]

    # Creamos la figura y el axis
    fig, (ax1, ax2) = plt.subplots(1, 2)

    # Realizamos el grafico
    label_name = r'$\hat{\Phi}^N_{u_{Bolus}}(\omega), \gamma = ' + gamma_buff
+ '$'
    ax1.loglog(U_bolo, color='#9373c8', label=r'Periodogram')
    ax1.loglog(abs(phi_buff), color='#124569', lineWidth=2, label=label_name)

    ax2.semilogy(U_bolo, color='#9373c8', label=r'Periodogram')
    ax2.semilogy(abs(phi_buff), color='#124569', lineWidth=2, label=label_name
)

    # Configuramos los parametros
    ax1.grid(True, which='both')
    ax1.set_ylim([10 ** (-3), 10 ** 1])
    ax1.set_xlim([10 ** (-6), max(U_bolo.index)])
    ax1.legend(fancybox=True, shadow=True)

    ax2.grid(True, which='both')
    ax2.set_ylim([10 ** (-3), 10 ** 1])
    ax2.set_xlim([10 ** (-6), max(U_bolo.index)])
    ax2.legend(fancybox=True, shadow=True, loc='lower left')

    ax1.set_ylabel(r'$\frac{Bolus Insulin^2}{Hz}$ [U$^2$/Hz]')
    ax1.set_xlabel(r'Frequency [Hz]')
    ax2.set_xlabel(r'Frequency [Hz]')

    x_size = 3 * 4.2
    y_size = 1 * x_size / 3
    fig.set_size_inches(x_size, y_size)
    plt.tight_layout()
```


Periodogram
$\hat{\Phi}^N_{u_{Bolus}}(\omega), \gamma = N/2$
Periodogram
$\hat{\Phi}^N_{u_{Bolus}}(\omega), \gamma = N/3$
Periodogram
$\hat{\Phi}^N_{u_{Bolus}}(\omega), \gamma = N/4$
Periodogram
$\hat{\Phi}^N_{u_{Bolus}}(\omega), \gamma = N/5$
Frequency [Hz]
$\frac{BolusInsulin^2}{Hz}$ [U^2/Hz]



Estmación de $\hat{G}_N(e^{i\omega})$, espectro de perturbación y espectro resiual

Estimación el espectro de G:

$$\hat{G}_N(e^{i\omega}) = \frac{\hat{\Phi}^N_{yu}(\omega)}{\hat{\Phi}^N_{u}(\omega)}$$

Estimación del espectro residual de V

$$\hat{\Phi}^N_{v}(\omega) = \hat{\Phi}^N_{y}(\omega) - \frac{\left|\hat{\Phi}^N_{yu}(\omega)\right|^2}{\hat{\Phi}^N_{u}(\omega)}$$

Espectro de coherencia:

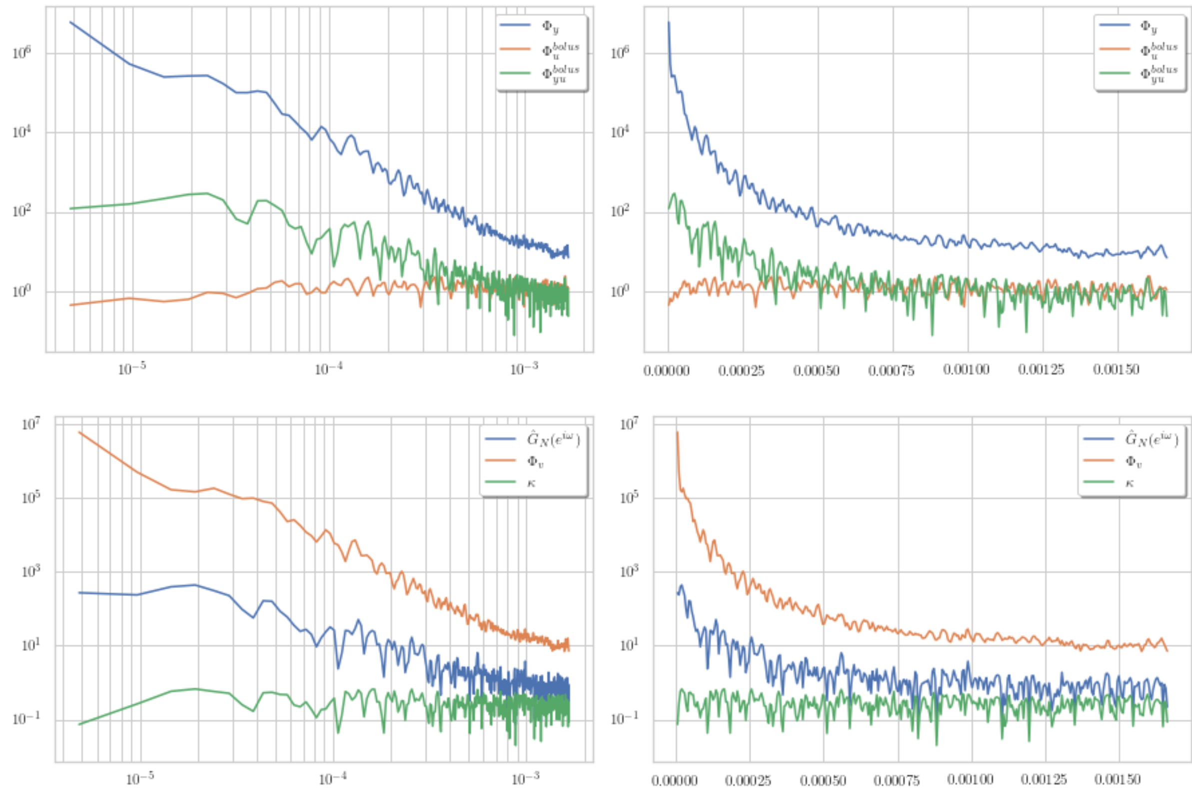
$$\kappa = \sqrt{\frac{\left|\hat{\Phi}^N_{yu}(\omega)\right|^2}{\hat{\Phi}^N_{y}(\omega)\hat{\Phi}^N_{u}(\omega)}}$$

In [62]:
```
# Obtenemos los datos
y = copy.copy(df['sensor_glucose'])
u_meal = copy.copy(df['meal'])
u_bolo = copy.copy(df['bolus_insulin'])

# Realizamos la interpolacion
y.interpolate(inplace=True, limit_direction='both')
y = y - y.mean()

u_meal.replace(np.nan, 0, inplace=True)
u_meal = u_meal - u_meal.mean()

u_bolo.replace(np.nan, 0, inplace=True)
u_meal = u_meal - u_meal.mean()
```

Estimación para bolo de insulina

In [279]:
```
N = len(y)
gamma = int(N/5)

R_u_bolo = np.correlate(u_bolo, u_bolo, mode='full') / N
phi_u_bolo = phi_X(R_u_bolo, gamma)

R_yu_bolo = np.correlate(y, u_bolo, mode='full') / N
phi_yu_bolo = phi_X(R_yu_bolo, gamma)

R_y = np.correlate(y, y, mode='full') / N
phi_y = phi_X(R_y, gamma)


G_hat = phi_yu_bolo / phi_u_bolo
phi_v = phi_y - (abs(phi_yu_bolo) ** 2) / phi_u_bolo
k = np.sqrt((abs(phi_yu_bolo) ** 2) / phi_u_bolo / phi_y)
```

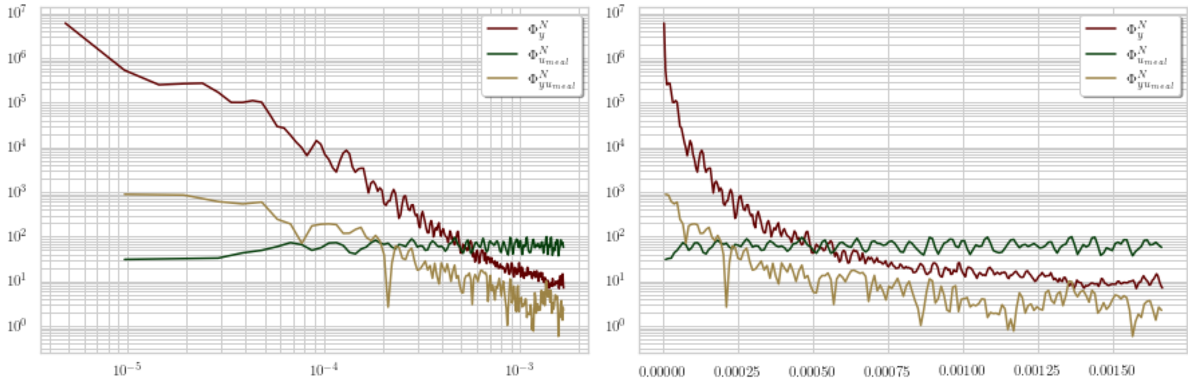
In [280]:
```
# Creamos la figura y el axis
fig, (ax1, ax2) = plt.subplots(1, 2)

# Realizamos el grafico
ax1.loglog(abs(phi_y), color='C0', label=r'$\Phi_y$')
ax1.loglog(abs(phi_u_bolo), color='C1', label=r'$\Phi_u^{bolus}$')
ax1.loglog(abs(phi_yu_bolo), color='C2', label=r'$\Phi_{yu}^{bolus}$')
ax2.semilogy(abs(phi_y), color='C0', label=r'$\Phi_y$')
ax2.semilogy(abs(phi_u_bolo), color='C1', label=r'$\Phi_u^{bolus}$')
ax2.semilogy(abs(phi_yu_bolo), color='C2', label=r'$\Phi_{yu}^{bolus}$')

# Configuramos los parametros
ax1.grid(True, which='both')
ax2.grid(True, which='both')
ax1.legend(fancybox=True, shadow=True)
ax2.legend(fancybox=True, shadow=True)

x_size = 3 * 4.2
y_size = 1 * x_size / 3
fig.set_size_inches(x_size, y_size)
plt.tight_layout()

# Creamos la figura y el axis
fig, (ax1, ax2) = plt.subplots(1, 2)

# Realizamos el grafico
ax1.loglog(abs(G_hat), color='C0', label=r'$\hat{G}_N(e^{i \omega})$')
ax1.loglog(abs(phi_v), color='C1', label=r'$\Phi_v$')
ax1.loglog(abs(k), color='C2', label=r'$\kappa$')
ax2.semilogy(abs(G_hat), color='C0', label=r'$\hat{G}_N(e^{i \omega})$')
ax2.semilogy(abs(phi_v), color='C1', label=r'$\Phi_v$')
ax2.semilogy(abs(k), color='C2', label=r'$\kappa$')

# Configuramos los parametros
ax1.grid(True,which='both')
ax2.grid(True,which='both')
ax1.legend(fancybox=True, shadow=True)
ax2.legend(fancybox=True, shadow=True)

x_size = 3 * 4.2
y_size = 1 * x_size / 3
fig.set_size_inches(x_size, y_size)
plt.tight_layout()
```

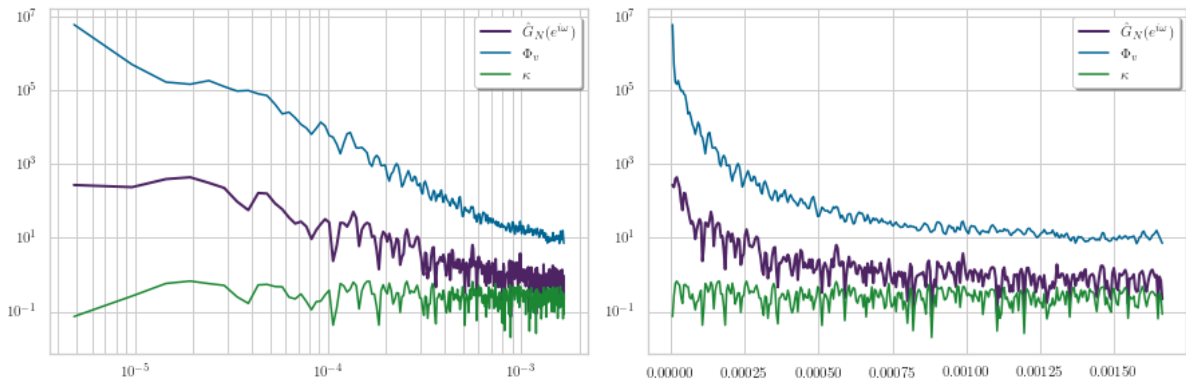


Estimación para la ingesta

In [129]:
```
N = len(y)
gamma = int(N/10)

R_u_meal = np.correlate(u_meal , u_meal , mode='full') / N
phi_u_meal  = phi_X(R_u_meal , gamma)

R_yu_meal = np.correlate(y, u_meal, mode='full') / N
phi_yu_meal = phi_X(R_yu_meal, gamma)

R_y = np.correlate(y, y, mode='full') / N
phi_y = phi_X(R_y, gamma)


G_hat = phi_yu_meal / phi_u_meal
phi_v = phi_y - (abs(phi_yu_meal) ** 2) / phi_u_meal
k = np.sqrt((abs(phi_yu_meal) ** 2) / phi_u_meal / phi_y)
```

In [295]:
```
# Creamos la figura y el axis
fig, (ax1, ax2) = plt.subplots(1, 2)

# Realizamos el grafico
ax1.loglog(abs(phi_y), color='#600000', label=r'$\Phi_y^N$')
ax1.loglog(abs(phi_u_meal), color='#053d0b', label=r'$\Phi_{u_{meal}}^N$')
ax1.loglog(abs(phi_yu_meal), color='#9c8345', label=r'$\Phi_{yu_{meal}}^N$')
ax2.semilogy(abs(phi_y), color='#600000', label=r'$\Phi_y^N$')
ax2.semilogy(abs(phi_u_meal), color='#053d0b', label=r'$\Phi_{u_{meal}}^N$')
ax2.semilogy(abs(phi_yu_meal), color='#9c8345', label=r'$\Phi_{yu_{meal}}^N$')

# Configuramos los parametros
ax1.grid(True, which='both')
ax2.grid(True, which='both')
ax1.legend(fancybox=True, shadow=True)
ax2.legend(fancybox=True, shadow=True)

x_size = 3 * 4.2
y_size = 1 * x_size / 3
fig.set_size_inches(x_size, y_size)
plt.tight_layout()

# Creamos la figura y el axis
fig, (ax3, ax4) = plt.subplots(1, 2)

# Realizamos el grafico
ax3.loglog(abs(G_hat), color='#4d2363', label=r'$\hat{G}_N(e^{i \omega})$', li
neWidth=2.0)
ax3.loglog(abs(phi_v), color='#006593', label=r'$\Phi_v$')
ax3.loglog(abs(k), color='#1b8733', label=r'$\kappa$')
ax4.semilogy(abs(G_hat), color='#4d2363', label=r'$\hat{G}_N(e^{i \omega})$',
lineWidth=2.0)
ax4.semilogy(abs(phi_v), color='#006593', label=r'$\Phi_v$')
ax4.semilogy(abs(k), color='#1b8733', label=r'$\kappa$')

# Configuramos los parametros
ax3.grid(True, which='both')
ax4.grid(True, which='both')
ax3.legend(fancybox=True, shadow=True)
ax4.legend(fancybox=True, shadow=True)

x_size = 3 * 4.2
y_size = 1 * x_size / 3
fig.set_size_inches(x_size, y_size)
plt.tight_layout()
```


Φ_y^N
$\Phi_{u_{meal}}^N$
$\Phi_{yu_{meal}}^N$


$\hat{G}_N(e^{i\omega})$
Φ_v
κ