

# Schwinger (Python)

A. Krivoi

18. August 2020

## Zusammenfassung

Beschreibung für das mit Python programmierte Schwinger-Programm, basierend auf dem Programm *Schwinger Version 2.0-4* der Physikalisch-Technischen Bundesanstalt, welches die Strahlung relativistischer Elektronen auf einer kreisförmigen Bahn berechnet. Die Berechnungsgrundlage ist eine Formel, die im Jahr 1949 von Julian Schwinger veröffentlicht wurde.


## 1 Systemvoraussetzung

Das Python-Schwinger-Programm ist plattformübergreifend. Voraussetzungen für die Nutzung sind:

- Python 3 (für die Programmierung wurde Python 3.7.4 genutzt)
- Programmbibliotheken:
  - NumPy
  - SciPy
  - Matplotlib

Das Programm kann ohne Entwicklungsumgebung ausgeführt werden. Für eine genauere Analyse der Ausgaben verschiedener Methoden empfiehlt sich die Python-IDE Spyder.

## 2 Installation des schwingerpy-packages

Das package `schwingerpy`  enthält eine Datei `__init__.py`, in der die Klasse **schwinger** definiert ist. Um neue Objekte dieser Klasse zu erstellen und die Methoden nutzen zu können, empfiehlt es sich, das package zu installieren. Dafür wird der Python Package Installer (pip) verwendet. Über die Konsole geht man in den Überordner `\schwingerpy`, in dem sich ein weiterer Ordner `\schwingerpy` und die `setup.py`-Datei befindet. Mit dem Befehl

```
pip install .
```

kann die Installation des packages gestartet werden. Auf der Konsole sollte anschließend `Successfully installed schwingerpy-0.1` erscheinen. Nun kann schwingerpy importiert werden. Falls die Installation des packages nicht erwünscht ist, kann die schwinger-Klasse auch einfach aus der `__init__.py`-Datei importiert werden. Dafür muss sich das Skript zum Ausführen jedoch im selben Ordner befinden.

## 3 Programm-Optionen

Nach der erfolgreichen Installation von schwingerpy lässt sich aus dem package die Klasse schwinger importieren:

```
from schwingerpy import schwinger
```

### 3.1 Neue Instanz der Klasse schwinger

Nun kann ein neues Objekt dieser Klasse für die anschließenden Schwinger-Berechnungen erzeugt werden. Jedes Objekt besitzt eine Reihe an Variablen, die in Tabelle 1 aufgeführt sind. Für alle Variablen gibt es Voreinstellungen, welche in der letzten Spalte der Tabelle aufgelistet sind. Wird ein neues Objekt mit einem beliebigen Namen, z.B. **test**, mit

```
test = schwinger()
```

erstellt, so erhält es die in der Tabelle angegebenen default-Werte. In diesem Fall wäre **test** ein Objekt für die Berechnung der spektralen Strahlungsleistung durch eine kreisförmige Blende mit einem Blendenradius  $r = 1$  mm in Abhängigkeit der Photonenenergie in eV.

Die Parameter können geändert werden, indem bei der Initialisierung des Objekts die gewünschten Werte in den Klammern angegeben werden. Wenn die Strahlungsleistung beispielsweise für eine rechteckige Blende mit den Abmaßen 2 x 2 mm für eine Elektronenenergie von 628 MeV und in Abhängigkeit der Wellenlänge in nm bestimmt werden soll, würde die Initialisierung so aussehen:

```
zweiterTest = schwinger(mode='rectangular', # Blendenform: rechteckig  
                        W=628,                # Elektronenenergie: 628 MeV  
                        aVert=2,              # vertik. Blendenweite: 2 mm  
                        aHoriz=2,             # horiz. Blendenweite: 2mm  
                        axUnit='nm')          # Einheit Skala: nm
```

Auf diese Art und Weise können bei Bedarf alle Variablen angepasst werden. Die Reihenfolge der Angabe spielt dabei keine Rolle.

| Variable       | Bedeutung   | Einheit          | Optionen   | default     |
|----------------|---|------------------|--|-------------|
| W              | Elektronenenergie   | MeV              | -  | 1719        |
| I              | Ringstrom   | mA               | -  | 100         |
| B              | magn. Flussdichte   | T                | -  | 1.3         |
| d              | Abstand<br>Quellpunkt - Blende                                | mm               | -  | 30000       |
| vOffset        | Vertikalversatz<br>der Blende                                 | mm               | -  | 0           |
| r              | Blendenradius   | mm               | nur relevant, wenn<br>mode='circular'  | 1           |
| aVert          | vertikale<br>Blendenweite                                     | mm               | nur relevant, wenn<br>mode='rectangular'   | 1           |
| aHoriz         | horizontale<br>Blendenweite                                   | mm               | nur relevant, wenn<br>mode='rectangular' bzw.<br>mode='vertical'                                   | 1           |
| sigmaPosition  | Standardabw. Ort<br>(Elektronen)                              | m                | nur relevant, wenn<br>emittance='yes'  | 0           |
| sigmaAngle     | Standardabw. Winkel<br>(Elektronen)                           | rad              | nur relevant, wenn<br>emittance='yes'  | 4e-6        |
| dlambda_dangle | Wellenlängenshift   | $\frac{nm}{rad}$ | nur relevant, wenn<br>mode='vertical'  | 0           |
| Np             | Anzahl Datenpunkte  | -                | mind. Np=2   | 500         |
| mode           | Modus für<br>Berechnung der<br>Strahlungsleistung             | -                | 'circular' (runde Blende)<br>'rectangular' (rechteckige Blende)<br>'vertical' (Vertikalverteilung) | 'circular'  |
| axUnit         | Einheit der Skala<br>für Berechnung der<br>Strahlungsleistung | -                | 'eV' (Photonenenergie / eV)<br>'nm' (Wellenlänge / nm)   | 'eV'        |
| axScale        | Verteilung der<br>Datenpunkte                                 | -                | 'linear'<br>'exponential'  | 'linear'    |
| axInput        | Achseneinteilung  | -                | 'increment' (Startwert, Inkrement)<br>'limits' (Startwert, Endwert)                                | 'increment' |
| abscissa       | Schrittweite<br>der Datenpunkte                               | eV<br>bzw. nm    | Angabe als Tupel<br>(s. axInput)   | (1,10)      |
| xVertical      | Wert für Berechnung<br>der Vertikalverteilung                 | eV<br>bzw. nm    | Einheit abhängig von axUnit  | 1           |
| emittance      | Berücksichtigung<br>der Emittanz                              | -                | 'no'<br>'yes'  | 'no'        |
| nStep          | Anzahl Integrations-<br>intervalle (Blende)                   | -                | gerade Anzahl wählen   | 150         |
| eStep          | Anzahl Emittanz-<br>stützstellen                              | -                | nur relevant, wenn emittance='yes'<br>gerade Anzahl wählen   | 150         |
| extent         | Ausdehnung Gauss-<br>verteilung (extent· $\Sigma_{Y'}$ )      | -                | nur relevant, wenn emittance='yes'   | 6           |

Tabelle 1: Instanzvariablen für Objekte der Klasse **schwinger**

Für die numerische Berechnung der Strahlungsleistung durch eine kreisförmige bzw. rechteckige Blende wird die Blendenfläche in mehrere Integrationsintervalle aufgeteilt. Die Anzahl der Intervalle wird hierbei mit der Variable **nStep** festgelegt. Falls die Emittanz berücksichtigt werden soll, indem **emittance='yes'** gesetzt wird, kommt bei der Strahlungsflussberechnung eine zusätzliche Integration hinzu. Dabei handelt es sich um eine Faltung der spektralen Strahlstärken über die effektive vertikale Strahldivergenz mit einer gaußförmigen Verteilung. Die Variable **extent** gibt dabei die Ausdehnung der Gaußverteilung an, also über wieviele  $\Sigma_Y$ -Breiten integriert wird. Mit **eStep** wird bestimmt, in wieviele Integrationsintervalle dieser Bereich aufgeteilt wird. Sowohl für **nStep**, als auch für **eStep**, wird eine gerade Anzahl an Intervallen erwartet. Bei Angabe einer ungeraden Zahl wird der Wert um 1 erhöht.

## 3.2 Methoden der Klasse **schwinger**

Die **schwinger**-Klasse besitzt verschiedene Methoden, um Berechnungen und Ausgaben durchzuführen. Diese werden im nachfolgenden Abschnitt vorgestellt.

### 3.2.1 **output()**

Mit der **output**-Methode kann nach der Erzeugung eines neuen Objekts die Berechnung der Strahlungsleistung mit den zuvor gewählten Eigenschaften gestartet werden. Die berechneten Daten werden in einem Textdokument ausgegeben. In Tabelle 2 sind die Parameter der Methode dargestellt.

| Variable  | Bedeutung  | Optionen   | default         |
|-----------|--|--|-----------------|
| filename  | Dateiname  | -  | 'schwingerTest' |
| plot      | Plot der berechneten Daten erstellen   | 'yes' (Plot wird erstellt)<br>'no' (Plot wird nicht erstellt)  | 'no'            |
| plotGraph | bestimmt, welche Komponenten der spektr. Strahlungsleistung geplottet werden | 'parallel' (parall. Komponente)<br>'senkrecht' (senkr. Komponente)<br>'gesamt' (spektraler Strahlungsfluss gesamt)<br>'all' (alle Graphen in einem Plot) | 'gesamt'        |

Tabelle 2: Parameter der **output**-Methode

Ein vollständiges Beispiel-Skript mit default-Einstellungen wäre:

```
from schwingerpy import schwinger

test = schwinger()
test.output()
```

Die Daten der Schwinger-Berechnung werden in diesem Fall als `schwingerTest.txt` gespeichert. Das Textdokument enthält einen Header mit den gewählten Eingangsparametern und die Ergebnisse der Schwinger-Berechnung. Es werden vier Spalten ausgegeben, welche die Skala, die Komponente der spektralen Strahlungsleistung mit paralleler bzw. senkrechter (und um 90° phasenverschoben) Polarisationsrichtung, sowie die gesamte spektrale Strahlungsleistung beinhalten. Bei Bedarf kann zur Visualisierung ein Plot erstellt werden, indem der Parameter `plot='yes'` gesetzt wird.

### 3.2.2 `calc_schwinger()`

Falls mit einer IDE gearbeitet wird und die Ergebnisse nicht direkt gespeichert werden sollen oder ein eigenes Python-Programm zur Weiterverarbeitung der Schwingerdaten erstellt wird, kann die `calc_schwinger`-Methode genutzt werden. Diese besitzt keine weiteren Parameter, die Rückgabe ist ein  $(N_p \times 4)$ -Array mit Abszissen-Achse, Strahlungsflusskomponenten parallel und senkrecht, sowie der gesamte spektrale Strahlungsfluss:

```
from schwingerpy import schwinger

test    = schwinger()
A       = test.calc_schwinger()  # (Np x 4)-Array mit Ergebnissen
                                   # der Schwinger-Berechnung
```

### 3.2.3 `simple_uncertainty()`

Die Funktion `simple_uncertainty()` berechnet näherungsweise den Einfluss der Messunsicherheiten auf die Unsicherheit der Strahlungsleistung mit der Methode der quadratischen Addition. Die Parameter sind in Tabelle 3 aufgelistet. Für die Einflussgrößen werden die Standardmessunsicherheiten in der entsprechenden Einheit angegeben. Die Methode gibt die absolute und relative Unsicherheit des gesamten spektralen Strahlungsflusses zurück speichert sie in einem Textdokument. Beispiele für die Verwendung der Methode finden sich in Kapitel 5.

| Variable        | Bedeutung   | default        |
|-----------------|---|----------------|
| u_W             | Standardmessunsicherheit<br>Elektronenenergie / MeV                                 | 0              |
| u_I             | Standardmessunsicherheit<br>Ringstrom / mA  | 0              |
| u_B             | Standardmessunsicherheit<br>magnetische Flussdichte / T                             | 0              |
| u_d             | Standardmessunsicherheit<br>Abstand Quellpunkt - Blende / mm                        | 0              |
| u_sigmaAngle    | Standardmessunsicherheit<br>Std-Abweichung Winkelverteilung<br>der Elektronen / rad | 0              |
| u_sigmaPosition | Standardmessunsicherheit<br>Std-Abweichung Ortsverteilung<br>der Elektronen / m     | 0              |
| u_vOffset       | Standardmessunsicherheit<br>Vertikalversatz Blende / mm                             | 0              |
| filename        | Dateiname   | 'schwingerMQA' |

Tabelle 3: Parameter der simple\_uncertainty-Methode

### 3.2.4 monte\_carlo()

Die Funktion nutzt die Monte-Carlo-Methode (MCM), um die Verteilungsfunktion der spektralen Strahlungsleistung numerisch zu nähern. Abhängig von den Wahrscheinlichkeitsverteilungen (Normal- bzw. Rechteckverteilung <sup>1</sup>, s. Tabelle 4) der Einflussgrößen werden Zufallswerte generiert und der Strahlungsfluss für diese zufälligen Wertepaare bestimmt. Anschließend wird der Erwartungswert und die Standardabweichung der resultierenden Verteilung berechnet. Da die MCM auf dem "Gesetz der großen Zahlen" basiert, ist eine große Anzahl an Samples notwendig, um eine möglichst genaue Approximation der Ausgangsverteilung zu erhalten. Daher sollte der Anwender selbst durch Testen bestimmen, welche Anzahl von Samples für den betrachteten Fall geeignet ist. Für eine grobe Abschätzung der Unsicherheiten bieten sich z.B. 500 Samples an. Bei genaueren Berechnungen sollten mindestens  $10^4$  Samples gewählt werden, wobei sich jedoch die Laufzeit des Programms deutlich verlängert. Die Monte-Carlo-Methode empfiehlt sich besonders, wenn eine Eingangsgröße die Messunsicherheit der Strahlungsleistung dominiert und asymmetrische Ausgangsverteilungen entstehen. Daher sollte der geschätzte Erwartungswert der Ausgangsverteilung immer mit dem Idealwert des Strahlungsflusses verglichen werden, beide werden zusammen mit der absoluten und relativen Unsicherheit des gesamten spektralen Strahlungsflusses in einem Textdokument gespeichert. Die Rückgaben der Methode sind die Abszisse, die Standardabweichung der gesamten spektralen Strahlungsleistung, sowie der geschätzte Erwartungswert und Median der Verteilungen.

<sup>1</sup>Standardmessunsicherheit  $u = \frac{\Delta a}{\sqrt{3}}$  mit der halben Intervallbreite  $\Delta a$  für Rechteckverteilungen

| Variable            | Bedeutung   | Optionen           | default        |
|---------------------|---|--------------------|----------------|
| u_W                 | Standardmessunsicherheit<br>Elektronenenergie / MeV                                 | -                  | 0              |
| distr_W             | Wahrscheinlichkeitsverteilung<br>Elektronenenergie                                  | 'normal'<br>'rect' | 'normal'       |
| u_I                 | Standardmessunsicherheit<br>Ringstrom / mA  | -                  | 0              |
| distr_I             | Wahrscheinlichkeitsverteilung<br>Ringstrom  | 'normal'<br>'rect' | 'normal'       |
| u_B                 | Standardmessunsicherheit<br>magnetische Flussdichte / T                             | -                  | 0              |
| distr_B             | Wahrscheinlichkeitsverteilung<br>magnetische Flussdichte                            | 'normal'<br>'rect' | 'rect'         |
| u_d                 | Standardmessunsicherheit<br>Abstand Quellpunkt - Blende / mm                        | -                  | 0              |
| distr_d             | Wahrscheinlichkeitsverteilung<br>Abstand Quellpunkt - Blende                        | 'normal'<br>'rect' | 'normal'       |
| u_sigmaAngle        | Standardmessunsicherheit<br>Std-Abweichung Winkelverteilung<br>der Elektronen / rad | -                  | 0              |
| distr_sigmaAngle    | Wahrscheinlichkeitsverteilung<br>Winkelverteilung                                   | 'normal'<br>'rect' | 'normal'       |
| u_sigmaPosition     | Standardmessunsicherheit<br>Std-Abweichung Ortsverteilung<br>der Elektronen / m     | -                  | 0              |
| distr_sigmaPosition | Wahrscheinlichkeitsverteilung<br>Ortsverteilung                                     | 'normal'<br>'rect' | 'normal'       |
| u_vOffset           | Standardmessunsicherheit<br>Vertikalversatz Blende / mm                             | -                  | 0              |
| distr_vOffset       | Wahrscheinlichkeitsverteilung<br>Vertikalversatz Blende                             | 'normal'<br>'rect' | 'normal'       |
| samples             | Anzahl der Samples<br>für Monte Carlo Methode                                       | -                  | 2000           |
| filename            | Dateiname   | -                  | 'schwingerMCM' |

Tabelle 4: Parameter der monte\_carlo-Methode

## 4 Ausführen des Python-Codes

Die einfachste Variante den Python-Code auszuführen, ist durch Doppelklick (Windows) bzw. Anklicken (Linux) des entsprechenden Icons. Dafür wird der Python-Code zunächst in ein Textdokument eingefügt und anschließend die Dateierweiterung von `.txt` zu `.py` geändert. Nun lässt sich das Programm direkt im Unterverzeichnis durch Klicken ausführen und die Textdateien mit den Ergebnissen werden im selben Ordner gespeichert.

Das Programm kann ebenfalls über die Eingabeaufforderung gestartet werden, indem man sich in das entsprechende Unterverzeichnis navigiert und die Datei (hierbei ist es egal, ob es sich um ein Textdokument oder ein Python File handelt) mit dem Befehl

```
python skript.py
```

aufruft.

Für die Erstellung eines Python Files kann auch eine Python-IDE, wie z.B. Spyder, genutzt werden. Dies empfiehlt sich besonders für die genauere Analyse und Weiterverarbeitung der Rückgaben der verschiedenen Methoden.

## 5 Beispiele

### 5.1 MLS Schwinger-Berechnungen

Das nachfolgende Skript berechnet den spektralen Strahlungsfluss durch eine kreisförmige Blende für einen Beispielparametersatz der MLS. Es wird die `output`-Methode verwendet, mit der die Ergebnisse in einem Textdokument gespeichert werden.

```
from schwingerpy import schwinger

# =====
# Berechnung der Strahlungsleistung durch eine kreisförmige Blende AP20
# in Abhängigkeit der Photonenenergie E / eV
# (Beispielparametersatz der MLS)
# =====

MLS = schwinger(W=600,           # Elektronenenergie: 600 MeV
                B=1.3,           # magnetische Flussdichte: 1.3 T
                I=100,            # Ringstrom: 100 mA
                d=15000,          # Abstand Blende: 15000 mm
                r=10,             # Blendenradius: 10 mm
                vOffset=0,        # Vertikalversatz Blende: 0 mm
                sigmaAngle=44e-6, # Standardabweichung vertikale
                                # Winkelvert. Elektronen: 44e-6 rad
                Np=300,           # Anzahl Datenpunkte: 300
                abscissa=(0.1, 10), # Startwert: 0.1 eV, Inkrement: 10 eV
                emittance='yes')  # Emittanz berücksichtigt

print('Wird berechnet ...')
# Erzeugung der Datei MLS_schwinger_ap20_mev600_t13.txt mit den
```



```
# Ergebnissen der Schwinger-Berechnung
MLS.output(filename='MLS_schwinger_ap20_mev600_t13', plot='yes')
```

Weiterhin wird der in Abbildung 1 gezeigte Plot erstellt, der die gesamte spektrale Strahlungsleistung in Abhängigkeit der Photonenenergie darstellt.

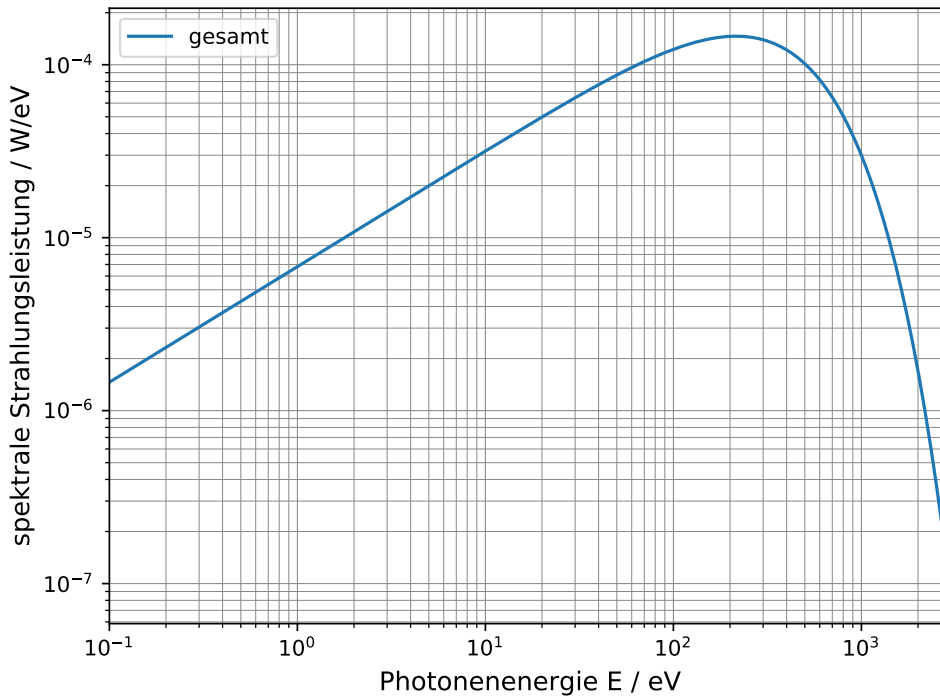


Abbildung 1:  
Berechnete Strahlungsleistung für  
Beispielparameter  
der MLS

Nach der Initialisierung eines Objekts können die Parameter nachträglich angepasst werden. Dies bietet sich z.B. an, wenn sich nur eine Eingangsgröße bei den Strahlungsflussberechnungen ändert. Dieses Skript erzeugt drei Textdokumente für verschiedene Blendenradien.

```
from schwingerpy import schwinger

# =====
# Berechnung der Strahlungsleistung für verschiedene Blendenradien
# in Abhängigkeit der Photonenenergie E / eV
# (Beispielparametersatz der MLS)
# =====

MLS = schwinger(W=600,           # Elektronenenergie: 600 MeV
                B=1.3,           # magnetische Flussdichte: 1.3 T
                I=100,           # Ringstrom: 100 mA
                d=15000,         # Abstand Blende: 15000 mm
                r=2.5,           # Blendenradius: 2.5 mm)
```

```

vOffset=0,          # Vertikalversatz Blende: 0 mm
sigmaAngle=44e-6,   # Standardabweichung vertikale
                    # Winkelvert. Elektronen: 44e-6 rad
Np=300,             # Anzahl Datenpunkte: 300
abscissa=(0.1, 10), # Startwert: 0.1 eV, Inkrement: 10 eV
emittance='yes')    # Emittanz berücksichtigt

print('Wird berechnet ...')
MLS.output(filename='MLS_schwinger_ap5_mev600_t13')    # Ergebnisse AP5

MLS.r = 5          # Blendenradius
                  # auf 5mm setzen
MLS.output(filename='MLS_schwinger_ap10_mev600_t13')   # Ergebnisse AP10

MLS.r = 20          # Blendenradius
                  # auf 20mm setzen
MLS.output(filename='MLS_schwinger_ap40_mev600_t13')   # Ergebnisse AP40

```

## 5.2 MLS - Vertikalverteilung

Wenn bei der Initialisierung des Objekts `mode='vertical'` gesetzt wird, kann die Vertikalverteilung der Synchrotronstrahlung für eine Photonenenergie bzw. Wellenlänge berechnet werden. Für die Abszissenachse werden in diesem Fall Werte für den vertikalen Winkel in rad angegeben. Ein beispielhaftes Skript ist nachfolgend für die MLS gezeigt.

```

from schwingerpy import schwinger

# =====
# Berechnung der Vertikalverteilung der Synchrotronstrahlung
# für die Photonenenergie E = 3 eV (Beispiel MLS)
# =====

MLS = schwinger(W=600,          # Elektronenenergie: 600 MeV
                B=1.3,          # magnetische Flussdichte: 1.3 T
                I=100,          # Ringstrom: 100 mA
                d=15000,        # Abstand Blende: 15000 mm
                vOffset=0,      # Vertikalversatz Blende: 0 mm
                sigmaAngle=44e-6, # Standardabweichung vertikale
                                # Winkelvert. Elektronen: 44e-6rad
                emittance='yes', # Emittanz berücksichtigt
                aHoriz=5,        # horizontale Blendenweite: 5mm
                mode='vertical', # Vertikalverteilung
                xVertical=3,     # Berechnung für E = 3 eV

```

```

axInput='limits',          # Angabe von Start- und Endwert
abscissa=(-1e-2, 1e-2),   # Startwert: -1e-2 rad
                             # Endwert: 1e-2 rad
Np=500)                    # Anzahl Datenpunkte: 500

print('Wird berechnet ...')
# Erzeugung der Datei MLS_vertical_distr_ev3.txt mit den
# Ergebnissen der berechneten Vertikalverteilung
MLS.output(filename='MLS_vertical_distr_ev3', plot='yes', plotGraph='all')

```

Da zusätzlich die Plot-Funktion aktiviert und `plotGraph='all'` gesetzt ist, wird die gesamte winkelabhängige spektrale Strahlungsleistung, sowie die der Komponenten mit paralleler und senkrechter Polarisationsrichtung dargestellt (s. Abb. 2).

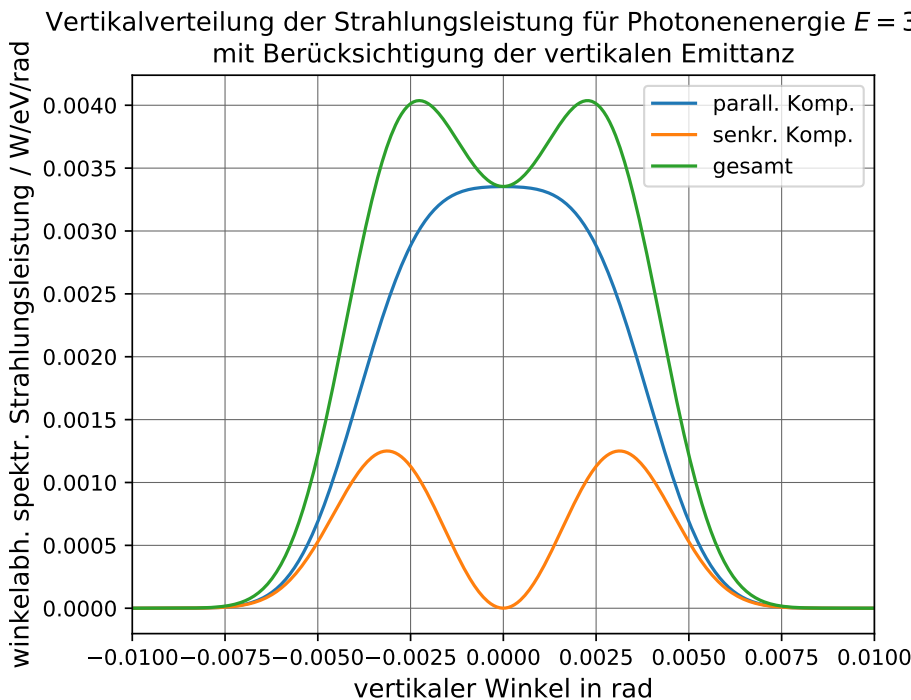


Abbildung 2:  
Berechnete  
Vertikalverteilung  
der Synchrotron-  
strahlung für  
Beispielparameter  
der MLS

### 5.3 MLS - Messunsicherheitsbetrachtung

Skript für die Unsicherheitsbestimmung mit der Methode der quadratischen Addition und der Monte-Carlo-Methode für Beispielparameter der MLS für vier Photonenenergien.

```

from schwingerpy import schwinger

# =====
# Einfluss der Messunsicherheiten verschiedener Größen auf die

```

```

# Unsicherheit der spektralen Strahlungsleistung durch eine
# kreisförmige Blende (AP20) in Abhängigkeit der Photonenenergie E / eV
# (Beispielparametersatz der MLS)
# =====

MLS = schwinger(W=600,          # Elektronenenergie: 600 MeV
                B=1.3,          # magnetische Flussdichte: 1.3 T
                I=100,          # Ringstrom: 100 mA
                d=15000,        # Abstand Blende: 15000 mm
                r=10,           # Blendenradius: 2.5 mm
                vOffset=0,      # Vertikalversatz Blende: 0 mm
                sigmaAngle=44e-6, # Standardabweichung vertikale
                                # Winkelvert. Elektronen: 44e-6rad
                emittance='yes', # Emittanz berücksichtigt
                axScale='exponential', # Achseneinteilung log- aquidist.
                abscissa=(1, 10), # Startwert: 1 eV,
                                # Multiplikator: 10 eV
                Np=4)           # Anzahl Datenpunkte: 4

# =====
# Unsicherheitsbestimmung - Methode der quadratischen Addition
# mit Angabe der Standardmessunsicherheiten der Einflussgrößen
# =====

MLS.simple_uncertainty(u_W=0.06,          # Std-U Elektronen-
                      u_B= 0.00013,      # energie: 0.06 MeV
                      u_I=0.02,          # Std-U Magnetfeld: 0.00013 T
                      u_d=2,             # Std-U Ringstrom: 0.02 mA
                      u_vOffset=0.075,   # Std-U Abstand: 2 mm
                      u_sigmaAngle=9e-6, # Std-U Vertikalvers.: 0.075 mm
                      filename='MLS_MQA_ap20_mev600_t13')

print('Unsicherheit mit Methode der quadratischen Addition berechnet.')

# =====
# Unsicherheitsbestimmung - Monte Carlo Methode
# mit Angabe der Standardmessunsicherheiten der Einflussgrößen
# =====

print("""Nun geht es weiter mit der Monte Carlo Methode.
Dies kann einen Moment dauern ...""")

MLS.monte_carlo(u_W=0.06,          # Std-U Elektronenenergie: 0.06 MeV
                # default: normalverteilt

```

```

u_B= 0.00013,      # Std-U Magnetfeld: 0.00013 T
                    # default: rechteckverteilt
u_I=0.02,          # Std-U Ringstrom: 0.02 mA
                    # default: normalverteilt
u_d=2,             # Std-U Abstand: 2 mm
                    # default: normalverteilt
u_vOffset=0.075,   # Std-U Vertikalversatz: 0.075 mm
                    # default: normalverteilt
u_sigmaAngle=9e-6, # Std-U Winkeldivergenz: 9e-6
                    # default: normalverteilt
samples=500,       # Anzahl Samples: 500
filename='MLS_MCM_ap20_mev600_t13')

```

## 5.4 BESSY II Schwinger-Berechnungen

Das nachfolgende Skript berechnet den spektralen Strahlungsfluss durch eine rechteckige Blende für einen Beispielparametersatz von BESSY II.

```

from schwingerpy import schwinger

# =====
# Berechnung der Strahlungsleistung durch eine rechteckige Blende 5x5 mm
# in Abhängigkeit der Wellenlänge lambda / nm
# (Beispielparametersatz für BESSY II)
# =====

BESSY2 = schwinger(W=1718.6,      # Elektronenenergie: 1718.6 MeV
                  B=1.29932,      # magn. Flussdichte: 1.29932 T
                  I=10,           # Ringstrom: 10 mA
                  d=30000,        # Abstand Blende: 30000 mm
                  vOffset=0,      # Vertikalversatz Blende: 0 mm
                  sigmaAngle=3.5e-6, # Standardabw. vert. Winkelvert.
                                  # Elektronen: 3.5e-6 rad
                  mode='rectangular', # Blendenform: rechteckig
                  aVert=5,        # vertikale Blendenweite: 5 mm
                  aHoriz=5,       # horizontale Blendenweite: 5mm
                  emittance='yes', # Emittanz berücksichtigt
                  axUnit='nm',    # Einheit Skala: Wellenlänge/nm
                  axInput='limits', # Angabe von Start- und Endwert
                  axScale='exponential', # Achseneinteilung log-äquidist.
                  abscissa=(0.02, 100), # Startwert: 0.02 nm,
                                  # Endwert: 100 nm
                  Np=500)         # Anzahl Datenpunkte: 500

```

```
print('Wird berechnet ...')
# Erzeugung der Datei BESSY2_schwinger_5x5_mev17186_t129932.txt mit den
# Ergebnissen der Schwinger-Berechnung
BESSY2.output(filename='BESSY2_schwinger_5x5_mev17186_t129932',plot='yes')
```

Da bei der `output`-Methode `plot='yes'` gesetzt ist, wird außerdem die gesamte spektrale Strahlungsleistung in Abhängigkeit der Wellenlänge geplottet, wie in Abbildung 3 gezeigt.

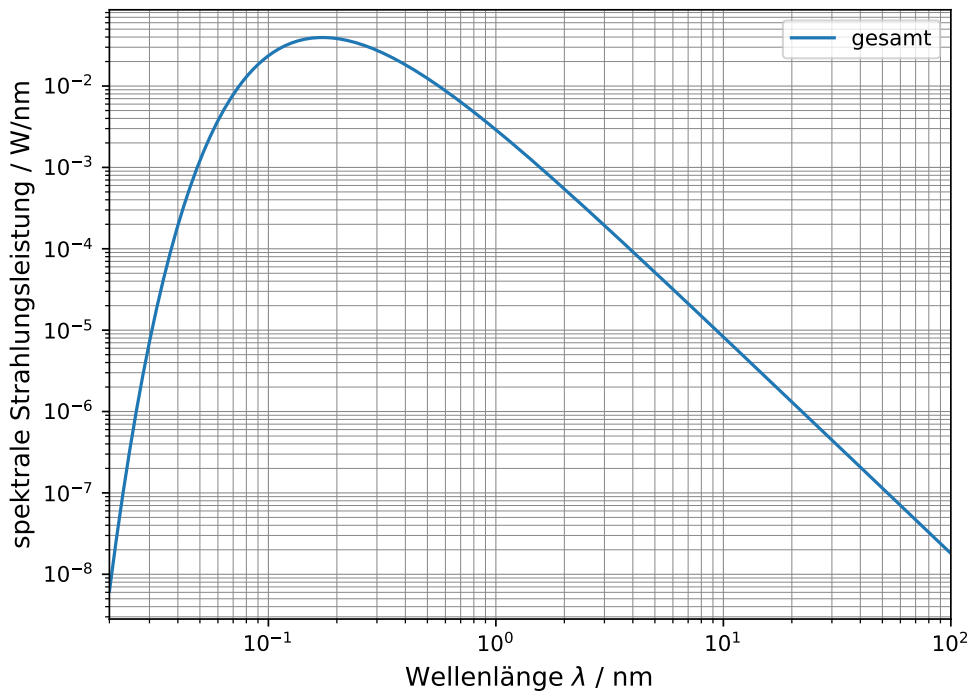


Abbildung 3:  
Berechnete Strahlungsleistung für Beispielparameter von BESSY II

## 5.5 BESSY II - Messunsicherheitsbetrachtung

Skript für die Unsicherheitsbestimmung mit der Methode der quadratischen Addition und der Monte-Carlo-Methode mit Beispielparametern von BESSY II für fünf Photonenenergien.

```
from schwingerpy import schwinger

# =====
# Einfluss der Messunsicherheiten verschiedener Größen auf die
# Unsicherheit der spektralen Strahlungsleistung durch eine
# rechteckige Blende (5x5 mm) in Abhängigkeit der Photonenenergie E / eV
# (Beispielparametersatz für BESSY II)
# =====
```

```

BESSY2 = schwinger(W=1718.6,          # Elektronenenergie: 1718.6 MeV
                  B=1.29932,          # magn. Flussdichte: 1.29932 T
                  I=10,               # Ringstrom: 10 mA
                  d=30000,            # Abstand Blende: 30000 mm
                  vOffset=0,          # Vertikalversatz Blende: 0 mm
                  sigmaAngle=3.5e-6,  # Standardabw. vert. Winkelvert.
                                      # Elektronen: 3.5e-6 rad
                  mode='rectangular',  # Blendenform: rechteckig
                  aVert=5,            # vertikale Blendenweite
                  aHoriz=5,           # horizontale Blendenweite
                  emittance='yes',     # Emittanz berücksichtigt
                  axScale='exponential', # Achseneinteilung log-äquidist.
                  abscissa=(3, 10),    # Startwert: 3 eV,
                                      # Multiplikator: 10 eV
                                      # Anzahl Datenpunkte: 5
                  Np=5)

# =====
# Unsicherheitsbestimmung - Methode der quadratischen Addition
# mit Angabe der Standardmessunsicherheiten der Einflussgrößen
# =====

BESSY2.simple_uncertainty(u_W=0.06,    # Std-U Elektronen-
                        # energie: 0.06 MeV
                        u_B= 0.00012,  # Std-U Magnetf.: 0.00012 T
                        u_I=0.002,      # Std-U Ringstrom: 0.002 mA
                        u_d=2,           # Std-U Abstand: 2 mm
                        u_vOffset=0.06,  # Std-U Vertikal-
                                      # versatz: 0.0 6mm
                        u_sigmaAngle=0.7e-6, # Std-U Winkeldiv.: 0.7e-6
                        filename='BESSY2_MQA_5x5_mev17186_t129932')

print('Unsicherheit mit Methode der quadratischen Addition berechnet.')

# =====
# Unsicherheitsbestimmung - Monte Carlo Methode
# mit Angabe der Standardmessunsicherheiten der Einflussgrößen
# =====

print("""Nun geht es weiter mit der Monte Carlo Methode.
Dies kann einen Moment dauern ...""")

BESSY2.monte_carlo(u_W=0.06,          # Std-U Elek.-Energie: 0.06 MeV
                  # default: normalverteilt
                  u_B=0.00012,        # Std-U Magnetfeld: 0.00013 T
                                      # default: rechteckverteilt

```

```
u_I=0.002,          # Std-U Ringstrom: 0.002 mA
                    # default: normalverteilt
u_d=2,              # Std-U Abstand: 2 mm
                    # default: normalverteilt
u_vOffset=0.06,     # Std-U Vertikalversatz: 0.06 mm
                    # default: normalverteilt
u_sigmaAngle=0.7e-6, # Std-U Winkeldivergenz: 0.7e-6
                    # default: normalverteilt
samples=500,        # Anzahl Samples: 500
filename='BESSY2_MCM_5x5_mev17186_t129932')
```