Simulation des radioaktiven Zerfalls

Veit Berger

T³ DEUTSCHLAND

Teachers Teaching with Technology™

Einleitung

Alljährlich gedenken wir am 7. November des Geburtstags der Physikerin und Chemikerin Marie Curie. Die gebürtige Polin entdeckte gemeinsam mit ihrem Ehemann Pierre Curie die Elemente Polonium und Radium und war maßgeblich an der Erforschung der radioaktiven Strahlung beteiligt. Auf sie geht der Begriff „Radioaktivität“ zurück. Trotz vieler Hürden in einer männerdominierten Wissenschaftswelt und einiger persönlicher Schicksalsschläge überzeugte sie durch ihre Beharrlichkeit und ihre überragenden fachlichen Fähigkeiten. Zu Recht erhielt sie 1903 zusammen mit ihrem Ehemann und Henri Becquerel den Nobelpreis für Physik sowie 1911 den Nobelpreis für Chemie.[1]
Der nachfolgenden Beitrag soll an die bahnbrechenden Leistungen dieser außergewöhnliche Frau erinnern.
Mit einem einfachen Algorithmus wird der radioaktive Zerfall simuliert und visualisiert. Dabei werden die zur Verfügung stehenden TI-NSpire-Werkzeuge genutzt, um aus dieser Simulation vergleichbare kernphysikalische Größen wie die Zerfallskonstante und die Halbwertszeit zu gewinnen.

Der radioaktive Zerfall

Die Eigenschaft von Atomkernen, sich spontan in andere umzuwandeln, wobei Energie in Form von Teilchen oder Strahlung frei wird, nennt man **Radioaktivität**.
Die Ursache der Radioaktivität ist die **Instabilität der Kerne**, die jeweils durch ein spezielles Protonen-Neutronen-Verhältnis entsteht.

Die **Aktivität** beschreibt, wie schnell der Kernzerfall erfolgt. Dabei kann aber nicht vorhergesagt werden, zu welchem Zeitpunkt ein einzelner Kern zerfällt. Es gilt:

$$A(t) = -\frac{dN(t)}{dt} = \lambda \cdot N(t)\,;\, \lambda = konst. \qquad (1)$$

A … Aktivität in Bq = s^{-1}
λ … Zerfallskonstante in s^{-1}
N … Anzahl der instabilen Kerne

Als Lösung der Differentialgleichung (1) ergibt sich das **Zerfallsgesetz**:

$$N(t) = N_0 \cdot e^{-\lambda \cdot t} \qquad (2)$$

N_0 … Anzahl der instabilen Kerne zum Zeitpunkt t = 0

Die **Halbwertszeit** gibt an, nach welcher Zeit die Hälfte der ursprünglich vorhandenen instabilen Kerne zerfallen ist:

$$\frac{N_0}{2} = N_0 \cdot e^{-\lambda \cdot t_H}$$

t_H … Halbwertszeit in s^{-1}

$$\ln(\frac{1}{2}) = -\lambda \cdot t_H$$

$$t_H = \frac{\ln(2)}{\lambda} \qquad (3)$$

Die Halbwertszeit ist wie die Zerfallskonstante eine kernphysikalische Größe, die charakteristisch für ein radioaktives Element und unabhängig von äußeren Einflüssen ist.

Der Algorithmus

Alle Zellen einer N x N – Matrix sind mit einem Farbewert belegt.
Zufällig wird nun eine Spalten- und eine Zeilenkoordinate ausgewählt. Falls die entsprechende Zelle diesen Farbwert besitzt, wird dieser gelöscht. Anderenfalls erfolgt keine Änderung. Die zufällige Auswahl wird so oft wiederholt, bis die Farbwerte aller Zellen zurückgesetzt wurden.

Die nachfolgenden Abbildungen illustrieren diesen Algorithmus.

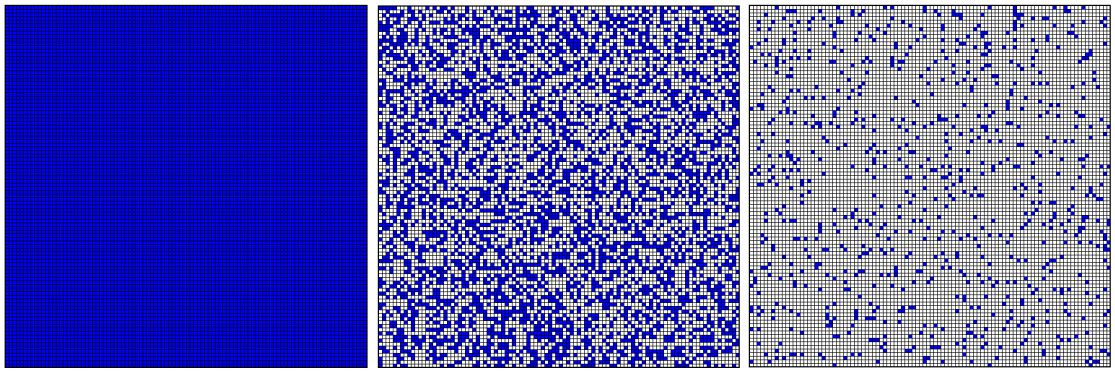

Abbildung 1: Veranschaulichung des Simulationsalgorithmus zum radioaktiven Zerfall

Ein erstes Python-Programm

Zunächst soll die Simulation und deren Auswertung ohne Visualisierung mit dem Programm **`Radioaktiver Zerfall1.tns`** implementiert werden. Der entsprechende Quelltext wird in der nachfolgenden Übersicht kurz kommentiert:

```
from ti_system import *
from random import randint
from time import ticks_ms, sleep_ms

N = 100; step =500

def simulation(N, step):
  i0 = N ** 2; i = i0
  t_list = [0]; n_list = [i]
  matrix = [[1 for x in range(N)]
              for y in range(N)]
  t0 = ticks_ms()
  while i > 0 and get_key() != "esc":
    x = randint(0, N - 1)
    y = randint(0, N - 1)
    if matrix[x][y] == 1:
      matrix[x][y] = 0
      i0 = i; i -= 1
    if i0 > i and i > 0 and i % step == 0:
      t = ticks_ms() - t0
      t_list.append(t / 1000)
      n_list.append(i)
      i0 = i
    sleep_ms(1)
  store_list("t", t_list)
  store_list("n", n_list)
  return (i)
```

Einbindung der erforderlichen Bibliotheken,

Festlegung der Dimension der Matrix sowie der Schrittweite der Messwerterfassung,

Definition der t- bzw. N-Liste mit den Startwerten t = 0 und N,
Belegung einer N x N - Matrix mit 1-Werten,

Festlegen der Startzeit
Schleife mit vorzeitigem Abbruch mit **<esc>**,
zufällige Wahl einer Spalten- bzw. Zeilen-Koordinate,

falls der Wert an dieser Stelle 1 ist, dann überschreibe ihn mit 0 und dekrementiere den Zähler i um 1,

falls ein Vielfaches der Schrittweite erreicht ist, bestimme die aktuelle Systemzeit und ergänze die Listen mit den aktuellen Messwerten,

Vereinbarung einer kurzen Pause,

Übertragung der t- bzw. N-Liste in TI-NSpire-Listen,
Rückgabe der verbleibenden Anzahl i,

Die Funktion **`simulation`** wird nun mit den aktuellen Parametern **`N`** und **`stepp`** aufgerufen:

```
clear_history()
print("Start der Simulation / Abbruch mit <esc>")
print("Anzahl der instabilen Kerne: ", simulation(N, step))
print("Simulation beendet!")
```

Durch die Nutzung von zwei Zählvariablen **`i0`** und **`i`** wird eine mögliche Mehrfacherfassung eines N(t)-Wertepaares unterdrückt.

Mit der Messwertübertragung in TI-NSpire-Listen stehen nun umfassende Auswertungs- und Analysefunktionen zur Verfügung.
Zunächst soll die N(t)-Abhängigkeit mit dem Statistikwerkzeug in der TI-Nspire-Umgebung grafisch dargestellt werden. Die exponentielle Regression bestätigt prinzipiell das Zerfallsgesetz (2) als Lösung der Differentialgleichung (1):

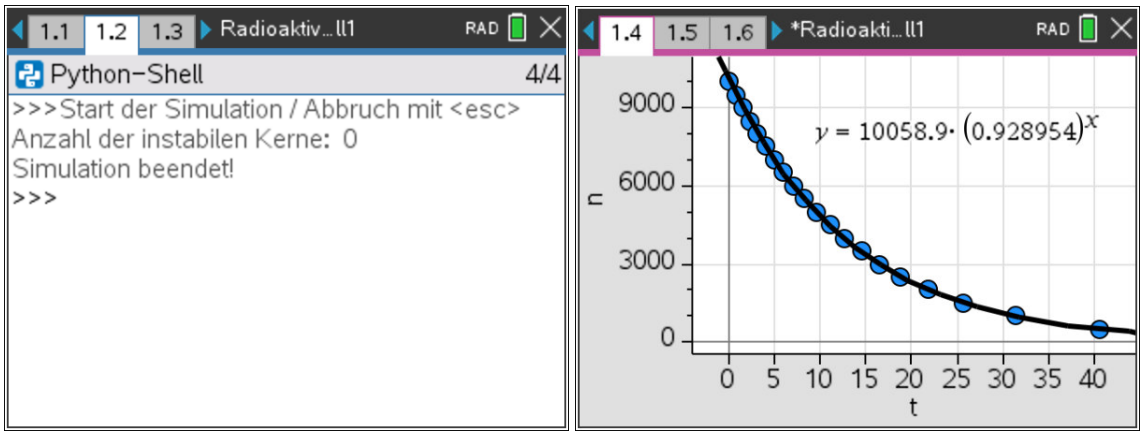

Abbildung 2: Simulation mit N(t)-Abhängigkeit und exponentieller Regression

In Analogie zum radioaktiven Zerfall sollen nun die Zerfallskonstante und die Halbwertszeit dieser Simulation bestimmt werden. Dazu werden die Messwerte für N sowie die Gleichung zum Zerfallsgesetz logarithmiert. Anschließend ist eine lineare Regression möglich:

$$\ln[N(t)] = -\lambda \cdot t + \ln(N_0) \qquad (4)$$

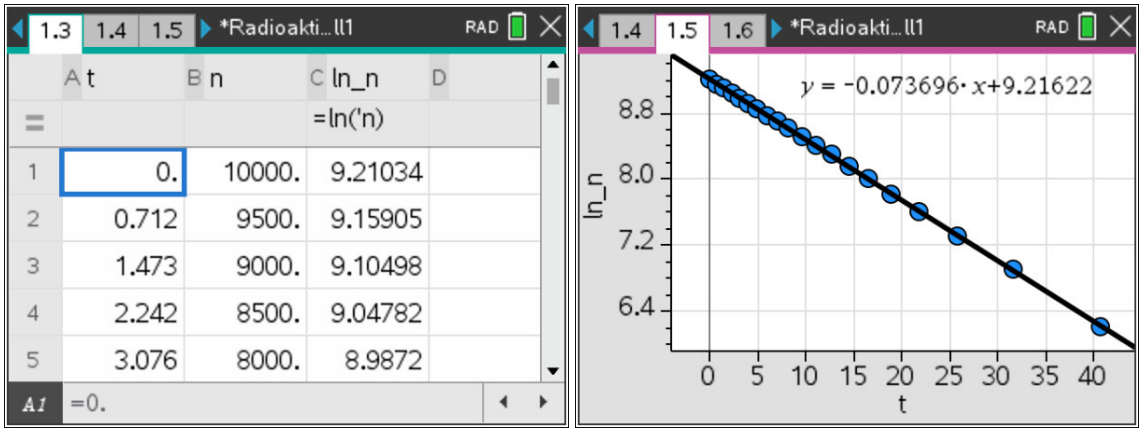

Abbildung 3: Wertetabelle und logarithmische Darstellung mit linearer Regression

Zunächst soll die Anzahl der instabilen Kerne zum Zeitpunkt t = 0 rechnerisch überprüft werden.
Die Zerfallskonstante λ wird direkt aus dem (negativen) Anstieg der Regressionsgeraden in der logarithmischen Darstellung ermittelt. Die Halbwertszeit t_H kann nun aus dieser Zerfallskonstante mit der Gleichung (3) berechnet werden.

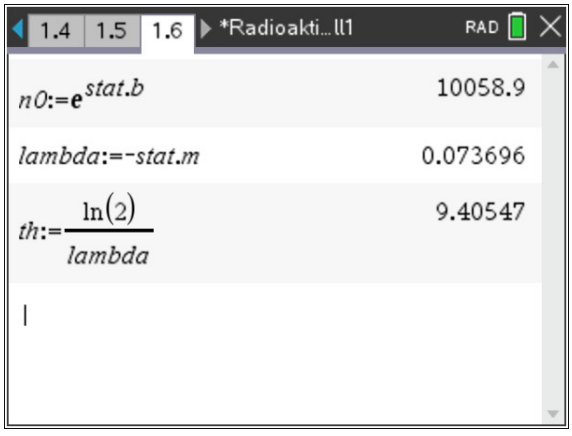

Abbildung 4: Auswertung der Simulation

Es soll noch angemerkt werden, dass der Gebrauch der Begriffe instabiler Kern, Zerfallskonstante bzw. Halbwertszeit in diesem Zusammenhang nicht ganz korrekt ist, denn die berechneten Werte sind an einfaches Simulationsmodell gebunden und nicht etwa die realen Größen eines speziellen radioaktiven Isotops.
Weiterhin sind die berechneten Werte sehr stark von den Geräteeigenschaften abhängig. So wird die Halbwertszeit auf einem Handheld einen anderen Wert annehmen, als unter sonst gleichen Bedingungen auf einem PC oder Notebook.

Simulation mit Visualisierung

Natürlich ist es wünschenswert, den beschriebenen Algorithmus an eine ähnliche Darstellung zu binden, wie sie in Abbildung 1 zu sehen ist.
Das Programm **`Radioaktiver Zerfall2.tns`** realisiert diese Visualisierung, indem es die Objektklasse **`screen`** nutzt, die für das Umschalten in den Grafikmodus sowie die textuelle Ein- und Ausgabe darin zuständig ist:

```
from classes import *
my_screen = screen("Simulation des radioaktiven Zerfalls")
```

Im nachfolgenden Programmausschnitt wird eine grafische N x N – Matrix mit dem Farbwert Blau belegt. Die globale Variable N kann in dem Intervall 25 ≤ N ≤ 100 in der entsprechenden Programmzeile variiert werden:

```
N = 100; step = 500
x0 = (310 - N) // 2; y0 = (206 - N) // 2

# --Startbildschirm--
rgb = (0, 0, 255); set_color(rgb)
my_screen.set_color_rgb(rgb)
draw_rect(x0 - 1, y0 - 1, N + 1, N + 1)
fill_rect(x0, y0, N, N)
```

In der Simulation werden nun gegebenenfalls sowohl Zahlen- als auch Farbwerte geändert: Zur weiteren Auswertung erfolgt auch hier Konvertierung der Messwerte in TI-NSpire-Listen.

```
# --Simulation--
i0 = N ** 2; i = i0
t_list = [0]; n_list = [i]
matrix = [[1 for x in range(N)] for y in range(N)]
rgb = (255, 255, 255); set_color(rgb)
my_screen.set_color_rgb(rgb)
t0 = ticks_ms()
while i > 0 and get_key() != "esc":
  x = randint(x0, x0 + N - 1)
  y = randint(y0, y0 + N - 1)
  t = (ticks_ms() - t0) / 1000
  if matrix[x - x0 - 1][y - y0 - 1] == 1:
    matrix[x - x0 - 1][y - y0 - 1] = 0
    plot_xy(x, y, 7)
    i0 = i; i -= 1
  if i0 > i and i > 0 and  i % step == 0:
    t_list.append(t)
    n_list.append(i)
    i0 = i
  my_screen.get_value(3, 13, "t = %1d s" % t)
  my_screen.get_value(11, 13, "N = %1d" % i)
  paint_buffer()
store_list("t", t_list)
store_list("n", n_list)
```

Zu beachten ist, dass mit der Visualisierung im Grafikmodus eine deutliche Verzögerung im Programmablauf und damit eine Zunahme der Halbwertszeit erwartet werden müssen.
Ein vorzeitiger Programmabbruch mit **<esc>** lässt dennoch die bereits beschriebenen Auswertungen in der TI-NSpire-Umgebung zu.
Die folgenden Abbildungen zeigen zwei Momentaufnahmen der grafischen Ausgabe.

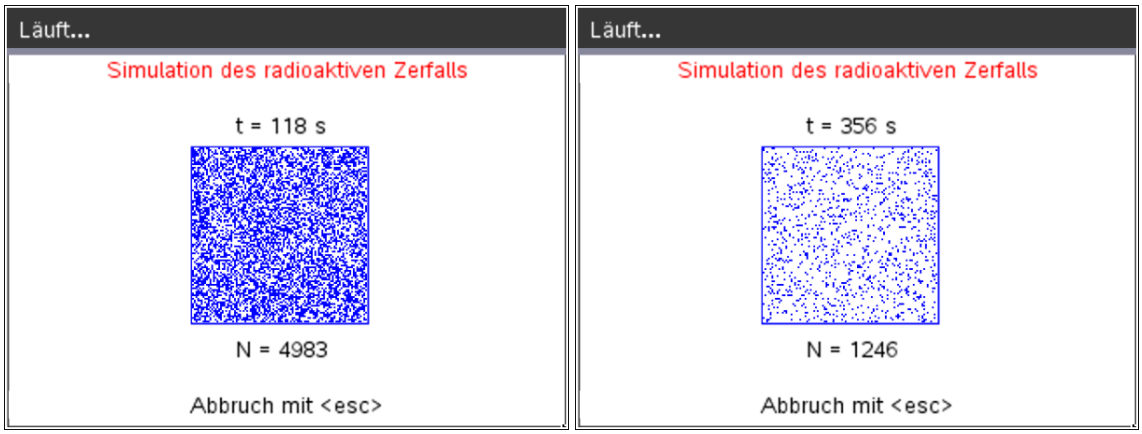

Abbildung 5: Programmablauf zur Simulation mit Visualisierung

Zusammenfassung

Der Thematisierung des radioaktiven Zerfalls im Physikunterricht sollte unbedingt der historische Einblick zur Entdeckung der Radioaktivität vorangestellt werden. Zunächst zufällig durch Henri Becquerel entdeckt, war es den systematischen experimentellen Untersuchungen von Marie und Pierre Curie zu verdanken, dass dieses Phänomen einzelnen bekannten bzw. neu entdeckten chemischen Elementen zuzuordnen ist. Eine Einordnung als kernphysikalische Erscheinung konnte erst später mit den besseren Vorstellungen zum Atomaufbau vorgenommen werden.[2]
Vor dem Einsatz der Simulation im Unterricht muss den Schülerinnen und Schülern klar sein, dass es sich dabei um ein sehr einfaches Modell zur Veranschaulichung handelt, das mit der Physik des Kernzerfalls nur bedingt vergleichbar ist. Schon die vergleichsweise geringe Anzahl der simulierten Umwandlungen hat mit der tatsächlichen Anzahl instabiler Kerne in der Natur nichts zu tun. Dennoch ist die Simulation geeignet, Analogien zum Kernzerfall herzustellen und daran Analyse- und Auswertmethoden zu entwickeln, die auf reale Messwerte anwendbar sind.

Quellen:

[1] https://de.wikipedia.org/wiki/Marie_Curie
[2] https://de.wikipedia.org/wiki/Kernphysik