Transformada Z e sua inversa em Python

Estudante
Bruno Lima Queiroz Santos

27 de junho de 2020

Sumário

1 Transformada Z pelo Python

A transformação Z analítica não existe até então em Python, nem em Octave, cujo pacote simbólico é baseado no pacote sympy do Python, embora muitas outras existam. Por isso, construiu-se uma função que pode ser desenvolvida e gera o resultado desejado em Python.

a) Cabeçalho

Em um formato Ipython-Notebook, usou-se o seguinte cabeçalho:

[1]

```
from IPython.display import (Image,
                             display,
                             Latex)
from sympy import (symbols,
                   latex,
                   DiracDelta,
                   Heaviside,
                   solve_poly_inequality,
                   Poly,
                   oo,Sum,symbols,Add,
                   Piecewise,cos,sin,exp,log)

n=symbols('n',integer=True)
a , N, z =symbols('a, N, z')
r, _0 =symbols('r, _0', real=True)

def dlatex(arg):
  display(Latex(arg))
  return None

def u(n):
  return(Heaviside(n,1))
```

Listing 1: Cabeçalho para a função ztrans e o seu uso.Código em ztrans_1.Py

b) Função ztrans

[2]

```
def zztrans(x_n, args):
  z=symbols('z')
  x_t, u_t = x_n.as_independent(args)
  arg_1=u_t.args[0] # Linear equation (polinomial)
  n_t=u_t.free_symbols # N is the only one variable.
  # The rest of all coefficients is constant.
  solve_p=solve_poly_inequality
  if(args == Heaviside ):
```

```
    interval=solve_p(Poly(arg_1, n_t, domain='ZZ'), '>')
    MIN=interval[0].left
    MAX=interval[0].right
  if(args == DiracDelta ):
    interval=solve_p(Poly(arg_1, n_t, domain='ZZ'), '==')
    MIN=interval[0].args[0]
    MAX=interval[0].args[0]
  return(Sum(x_t*(z**-n),(n_t,MIN,MAX)).doit())

def ztransf(x_n):
  if(x_n.has(Heaviside)):
    return(
        zztrans(x_n,Heaviside)
    )

  if(x_n.has(DiracDelta)):
    return(
        zztrans(x_n,DiracDelta)
    )
def ztrans(x_n):
  """
    Input is x[n]
  """
  ztransformation=0
  for x_term in Add.make_args(x_n):
    ztransformation+=ztransf(x_term)
  return (ztransformation.doit())#.simplify().simplify())
  #return (ztransformation.doit().simplify().simplify())
```

Listing 2: Código para a construção da função ztrans.Código em ztrans_2.Py

Depois, seguiu-se tabelando as fórmulas mais recorrentes, objetivo principal desse desenvolvimento.

c) Fórmulas

Fórmula 1

[3]

```
"""Fórmula 1"""


A=r"\text{Fórmula 1 :}"
dlatex(A)
print("")

A=r"\delta[n] \; \leftrightarrow \;"

A+=latex(ztrans(DiracDelta(n)))
dlatex(
    A
```

```
)
```

Listing 3: Fórmula 1.Código em ztrans_3.Py

O resultado foi o seguinte:

Fórmula 1 :

$$\delta[n] \; \leftrightarrow \; 1$$

Figura 1: Resultado do código ztrans_3.Py

Fórmula 2

[4]

```
"""Fórmula 2"""

A=r"\text{Fórmula 2 :}"
dlatex(A)
print("")

A=r"u[n] \; \leftrightarrow \;"

A+=latex(ztrans(
    u(n)
  ).simplify()
)
dlatex(
    A
)
```

Listing 4: Fórmula 2.Código em ztrans_4.Py

O resultado foi o seguinte:

Fórmula 2 :

$$u[n] \; \leftrightarrow \; \begin{cases} \frac{z}{z-1} & \text{for } \left|\frac{1}{z}\right| < 1 \\ \sum_{n=0}^{\infty} z^{-n} & \text{otherwise} \end{cases}$$

Figura 2: Resultado do código ztrans_3.Py

Fórmula 3

[5]

```
"""Fórmula 3"""
```

```
A=r"\text{Fórmula 3 :}"
dlatex(A)
print("")

A=r"-u[-n-1] \; \leftrightarrow \;"

A+=latex(ztrans(
    -u(-n-1)
  ).simplify()
)
dlatex(
    A
)
```

Listing 5: Fórmula 3.Código em ztrans_5.Py

O resultado foi o seguinte:

Fórmula 3 :

$$-u[-n-1] \; \leftrightarrow \; \begin{cases} \frac{z}{z-1} & \text{for } |z| < 1 \\ -\sum_{n=-\infty}^{-1} z^{-n} & \text{otherwise} \end{cases}$$

Figura 3: Resultado do código ztrans_5.Py

Fórmula 4

[6]

```
"""Fórmula 4"""

A=r"\text{Fórmula 4 :}"
dlatex(A)
print("")

A=r"\delta[n-5] \; \leftrightarrow \;"

A+=latex(ztrans(
    DiracDelta(n-5)
  )
)
dlatex(
    A
)
```

Listing 6: Fórmula 4.Código em ztrans_6.Py

O resultado foi o seguinte:

Fórmula 4 :

$$\delta[n-5] \;\leftrightarrow\; \frac{1}{z^5}$$

Figura 4: Resultado do código ztrans_6.Py

Fórmula 5

[7]

```
"""Fórmula 5"""

A=r"\text{Fórmula 5 :}"
dlatex(A)
print("")

A=r"a^{n}u[n] \; \leftrightarrow \;"

A+=latex(ztrans(
    a**n*u(n)
  ).simplify()
)
dlatex(
    A
)
```

Listing 7: Fórmula 5.Código em ztrans_7.Py

O resultado foi o seguinte:

Fórmula 5 :

$$a^n u[n] \;\leftrightarrow\; \begin{cases} \frac{z}{-a+z} & \text{for } \left|\frac{a}{z}\right| < 1 \\ \sum_{n=0}^{\infty} a^n z^{-n} & \text{otherwise} \end{cases}$$

Figura 5: Resultado do código ztrans_7.Py

Fórmula 6

[8]

```
"""Fórmula 6"""

A=r"\text{Fórmula 6 :}"
dlatex(A)
print("")

A=r"-a^{n}u[-n-1] \; \leftrightarrow \;"
```

```
A+=latex(ztrans(
    -a**n*u(-n-1)
  ).simplify().simplify()
)
dlatex(
    A
)
```

Listing 8: Fórmula 6.Código em ztrans_8.Py

O resultado foi o seguinte:

Fórmula 6 :

$$-a^n u[-n-1] \leftrightarrow \begin{cases} \frac{z}{-a+z} & \text{for } \left|\frac{z}{a}\right| < 1 \\ -\sum_{n=-\infty}^{-1} a^n z^{-n} & \text{otherwise} \end{cases}$$

Figura 6: Resultado do código ztrans_8.Py

Fórmula 7

[9]

```
"""Fórmula 7"""

A=r"\text{Fórmula 7 :}"
dlatex(A)
print("")

A=r"na^{n}u[n] \; \leftrightarrow \;"

A+=latex(ztrans(
    n*a**n*u(n)
  ).simplify()
)
dlatex(
    A
)
```

Listing 9: Fórmula 7.Código em ztrans_9.Py

O resultado foi o seguinte:

Fórmula 7 :

$$na^n u[n] \leftrightarrow \begin{cases} \frac{az}{(a-z)^2} & \text{for } \left|\frac{a}{z}\right| < 1 \\ \sum_{n=0}^{\infty} a^n n z^{-n} & \text{otherwise} \end{cases}$$

Figura 7: Resultado do código ztrans_9.Py

Fórmula 8

[10]

```
"""Fórmula 8"""

A=r"\text{Fórmula 8 :}"
dlatex(A)
print("")

A=r"-na^{n}u[-n-1] \; \leftrightarrow \;"

A+=latex(ztrans(
    -n*a**n*u(-n-1)
  ).simplify()
)
dlatex(
    A
)
```

Listing 10: Fórmula 8.Código em ztrans_10.Py

O resultado foi o seguinte:

Fórmula 8 :

$$-na^n u[-n-1] \leftrightarrow \begin{cases} \frac{az}{a^2-2az+z^2} & \text{for } \left|\frac{z}{a}\right| < 1 \\ -\sum_{n=-\infty}^{-1} a^n n z^{-n} & \text{otherwise} \end{cases}$$

Figura 8: Resultado do código ztrans_10.Py

Fórmula 9

[11]

```
"""Fórmula 9"""

A=r"\text{Fórmula 9 :}"
dlatex(A)
print("")

A=r"\cos(\omega_0 n)u[n] \; \leftrightarrow \;"

r_1=ztrans(
    (a**n +a**(-n))*u(n)*(1/2)
  ).subs(a,exp(1j *_0))
A+=latex((r_1).simplify())
dlatex(A)
```

Listing 11: Fórmula 9.Código em ztrans_11.Py

Obsvervação: _0 ↔ ω_0

O resultado foi o seguinte:

Fórmula 9 :

$$\cos(\omega_0 n)u[n] \;\leftrightarrow\; \begin{cases} 0.5z\left(\frac{e^{1.0i\omega_0}}{ze^{1.0i\omega_0}-1}+\frac{1}{z-e^{1.0i\omega_0}}\right) & \text{for } \left|\frac{1}{z}\right|<1 \\ 0.5\sum_{n=0}^{\infty} z^{-n}e^{-1.0in\omega_0}+0.5\sum_{n=0}^{\infty} z^{-n}e^{1.0in\omega_0} & \text{otherwise} \end{cases}$$

Figura 9: Resultado do código ztrans_11.Py

Fórmula 10

[12]

```
"""Fórmula 10"""

A=r"\text{Fórmula 10 :}"
dlatex(A)
print("")

A=r"\sin(\omega_0 n)u[n] \; \leftrightarrow \;"

r_1=ztrans(
    (a**n-a**(-n))*u(n)*(1/2j)
  ).subs(a,exp(1j *_0))
A+=latex((r_1).simplify())
dlatex(A)
```

Listing 12: Fórmula 10.Código em ztrans_12.Py

Obsvervação: _0 ↔ ω_0

O resultado foi o seguinte:

$$\sin(\omega_0 n)u[n] \;\leftrightarrow\; \begin{cases} 0.5iz\left(\frac{e^{1.0i\omega_0}}{ze^{1.0i\omega_0}-1}-\frac{1}{z-e^{1.0i\omega_0}}\right) & \text{for } \left|\frac{1}{z}\right|<1 \\ 0.5i\left(\sum_{n=0}^{\infty} z^{-n}e^{-1.0in\omega_0}-\sum_{n=0}^{\infty} z^{-n}e^{1.0in\omega_0}\right) & \text{otherwise} \end{cases}$$

Figura 10: Resultado do código ztrans_12.Py

Fórmula 11

[13]

```
"""Fórmula 11"""
```

```
A=r"\text{Fórmula 11 :}"
dlatex(A)
print("")

A=r"r^{n}\cos(\omega_0 n)u[n] \; \leftrightarrow \;"

r_1=ztrans(
    r**n*(a**n+a**(-n))*u(n)*(1/2)
  ).subs(a,exp(1j *_0))
A+=latex((r_1).simplify())
dlatex(A)
```

Listing 13: Fórmula 11.Código em ztrans_13.Py

Obsvervação: _0 ↔ ω_0

O resultado foi o seguinte:

Fórmula 11 :

$$r^n \cos(\omega_0 n)u[n] \;\leftrightarrow\; \begin{cases} 0.5z\left(\frac{1}{-re^{1.0i\omega_0}+z}+\frac{e^{1.0i\omega_0}}{-r+ze^{1.0i\omega_0}}\right) & \text{for } \left|\frac{r}{z}\right|<1 \\ 0.5\sum_{n=0}^{\infty} r^n z^{-n} e^{-1.0in\omega_0}+0.5\sum_{n=0}^{\infty} r^n z^{-n} e^{1.0in\omega_0} & \text{otherwise} \end{cases}$$

Figura 11: Resultado do código ztrans_13.Py

Fórmula 12

[14]

```
"""Fórmula 12"""

A=r"\text{Fórmula 12 :}"
dlatex(A)
print("")

A=r"r^{n}\sin(\omega_0 n)u[n] \; \leftrightarrow \;"

r_1=ztrans(
    r**n*(a**n-a**(-n))*u(n)*(1/2j)
  ).subs(a,exp(1j *_0))
A+=latex((r_1).simplify())
dlatex(A)
```

Listing 14: Fórmula 12.Código em ztrans_14.Py

Obsvervação: _0 ↔ ω_0

O resultado foi o seguinte:

$$r^n \sin(\omega_0 n)u[n] \;\leftrightarrow\; \begin{cases} 0.5iz\left(-\frac{1}{-re^{1.0i\omega_0}+z}+\frac{e^{1.0i\omega_0}}{-r+ze^{1.0i\omega_0}}\right) & \text{for } \left|\frac{r}{z}\right|<1 \\ 0.5i\left(\sum_{n=0}^{\infty} r^n z^{-n} e^{-1.0in\omega_0} - \sum_{n=0}^{\infty} r^n z^{-n} e^{1.0in\omega_0}\right) & \text{otherwise} \end{cases}$$

Figura 12: Resultado do código ztrans_14.Py

d) Tabela de fórmulas

Fórmula	x[n]	X(z)	Convergência	Comentários
1	$\delta[n]$	1	$\forall z$	
2	$u[n]$	$\frac{z}{z-1}$	$\left\|\frac{1}{z}\right\|<1$	
3	$-u[n-1]$	$\frac{z}{z-1}$	$\|z\|<1$	
4	$\delta[n-5]$	$\frac{1}{z^5}$	$\forall z$	$\forall m \in \mathbf{Z}$, além de $m=5$
5	$a^n u[n]$	$\frac{z}{-a+z}$	$\left\|\frac{a}{z}\right\|<1$	
6	$-a^n u[-n-1]$	$\frac{z}{-a+z}$	$\left\|\frac{z}{a}\right\|<1$	
7	$na^n u[n]$	$\frac{az}{(a-z)^2}$	$\left\|\frac{a}{z}\right\|<1$	
8	$-na^n u[-n-1]$	$\frac{az}{a^2-2az+z^2}$	$\left\|\frac{z}{a}\right\|<1$	
9	$\cos(\omega_0 n)u[n]$	$0.5z\left(\frac{e^{1.0i\omega_0}}{ze^{1.0i\omega_0}-1}+\frac{1}{z-e^{1.0i\omega_0}}\right)$	$\left\|\frac{1}{z}\right\|<1$	
10	$\sin(\omega_0 n)u[n]$	$0.5iz\left(\frac{e^{1.0i\omega_0}}{ze^{1.0i\omega_0}-1}-\frac{1}{z-e^{1.0i\omega_0}}\right)$	$\left\|\frac{1}{z}\right\|<1$	
11	$r^n\cos(\omega_0 n)u[n]$	$0.5z\left(\frac{1}{-re^{1.0i\omega_0}+z}+\frac{e^{1.0i\omega_0}}{-r+ze^{1.0i\omega_0}}\right)$	$\left\|\frac{r}{z}\right\|<1$	
12	$r^n\sin(\omega_0 n)u[n]$	$0.5iz\left(-\frac{1}{-re^{1.0i\omega_0}+z}+\frac{e^{1.0i\omega_0}}{-r+ze^{1.0i\omega_0}}\right)$	$\left\|\frac{r}{z}\right\|<1$	

Tabela 1: Tablela de transformações Z comuns e contempladas.

Obs: para maiores simplificações, pode-se utilizar .simplify() ao final das expressões simbólicas.

2 Transformada Z inversa pelo Python

A transformação inversa é mais delicada, sendo baseada em técnicas não tão elegantes para contornar as limitações da função integrate da library sympy. A transformação inversa construída aqui só funciona para as fórmulas de 5 até 8 se $a \in \mathbf{Z}$.

a) Cabeçalho

[1]

```
from IPython.display import (Image,
                             display,
                             Latex)
from sympy import (symbols,
                   latex,
                   DiracDelta,
                   Heaviside,
                   solve_poly_inequality,
                   Poly,
                   oo,Sum,Add,
                   Piecewise)

from sympy import (pi,
                   integrate,
                   cos,sin,Abs)

def dlatex(arg):
  display(Latex(arg))
  return None
```

Listing 15: Cabeçalho para a função iztrans e o seu uso.Código em iztrans_1.Py

[2]

```
def zztrans(x_n, args):
  z=symbols('z')
  x_t, u_t = x_n.as_independent(args)
  arg_1=u_t.args[0] # Linear equation (polinomial)
  n_t=u_t.free_symbols # N is the only one variable.
  # All the rest of coefficients are constant.
  solve_p=solve_poly_inequality
  if(args == Heaviside ):
    interval=solve_p(Poly(arg_1, n_t, domain='ZZ'), '>')
    MIN=interval[0].left
    MAX=interval[0].right
  if(args == DiracDelta ):
    interval=solve_p(Poly(arg_1, n_t, domain='ZZ'), '==')
    MIN=interval[0].args[0]
    MAX=interval[0].args[0]
  return(Sum(x_t*(z**-n),(n_t,MIN,MAX)).doit())

def ztransf(x_n):
```

```
  if(x_n.has(Heaviside)):
    return(
        zztrans(x_n,Heaviside)
    )

  if(x_n.has(DiracDelta)):
    return(
        zztrans(x_n,DiracDelta)
    )
def ztrans(x_n):
  """
    Input is x[n]
  """
  ztransformation=0
  for x_term in Add.make_args(x_n):
    ztransformation+=ztransf(x_term)
  return (ztransformation.doit())#.simplify().simplify())
  #return (ztransformation.doit().simplify().simplify())

def u(n):
  return(Heaviside(n,1))
```

Listing 16: Cabeçalho para a função iztrans e o seu uso.Código em iztrans_2.Py

b) Função iztrans

[3]

```
def iztrans(F,n,m=1.0):
  # m = |(z/a)|
  from sympy import symbols
  z,t=symbols('z,t')
  from numpy import pi as npi
  from sympy import pi as spi
  from sympy import (limit,
                     integrate)
  from sympy import (log,cos,sin)
  from numpy import log as nplog
  from numpy import cos as npcos
  from numpy import sin as npsin
  from numpy import exp as npexp
  from sympy import Heaviside

  func=F

 # GAMBIARRA
  """Corrigindo Bugs de integrate"""
  if(abs(m)<1):
    #n=n+1
    n=-1*n
    func=-F*z**(n-1)
  else:
    func=F*z**(n-1)
```

```
  m=1
  """Corrigindo Bugs de integrate"""
  # GAMBIARRA

  #func=func.subs(z,m*cos(t)+m*1j*sin(t))
  #func=func*(-sin(t)+1j*cos(t))*m
  integral_result=integrate(func,z)


  int_param=integral_result.subs(z,abs(m)*(cos(t)+1j*sin(t)) )
  result=(int_param.subs(t,npi)-int_param.subs(t,-npi)).simplify()
  result=(spi/npi)*result.simplify()
  result=result/(2*spi*1j).simplify()

  """Corrigindo Bugs de integrate"""
  if(abs(m)<1):
    result=result*Heaviside(-n-1,1)
  """Corrigindo Bugs de integrate"""

  return(result.simplify())
```

Listing 17: definição da função iztrans e o seu uso.Código em iztrans_3.Py

A função pode ser desenvolvida, estabelecendo-se a identificação dos inputs no contexto da tabela de transformação Z. O que foi criado foi a obtenção de $x[n]$ a partir da expressão integral de inversão. Além disso, é necessário atentar para o fator $m = |z/a|$ quando for necessário que seu valor seja menor que 1.

c) Fórmulas

Então, começou-se a testar as inversões.

[4]

```
z=symbols('z')
a=symbols('a')
from numpy import arange as nparange
from matplotlib.pyplot import stem,show,figure
```

Listing 18: definições iniciais. Código em iztrans_4.Py

[5]

```
title="""Simétrico da Fórmula 1"""
N=20
X_z=1.0
results=[ iztrans(X_z,i) for i in nparange(-N,N+1)]

fig = figure()
ax1 = fig.add_subplot(111)
ax1.set_ylabel('Valor',fontsize=15)
```

```
ax1.set_xlabel('N',fontsize=15)
ax1.set_title(title,fontsize=18)
stem(nparange(-N,N+1),results, use_line_collection=True,linefmt=None,
     markerfmt=None, basefmt=None)
show()
```

Listing 19: Simétrico da fórmula 1.Código em iztrans_5.Py

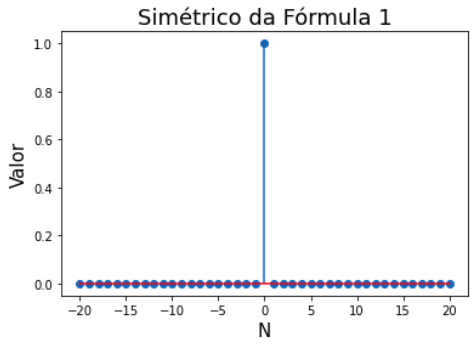

Figura 13: Resultado do código iztrans_5.Py

[6]

```
n=symbols('n')
print(ztrans(
    u(n)
  )
)
```

Listing 20: Aplicação de ztrans. Código em iztrans_6.Py

```
Piecewise((1/(1 - 1/z), Abs(1/z) < 1), (Sum(z**(-n), (n, 0, oo)), True))
```

Figura 14: Resultado do código iztrans_6.Py

[7]

```
title="""Simétrico da Fórmula 2"""
N=20
X_z=Piecewise((z/(z - 1), True), (Sum(z**(-n), (n, 0, oo)), True))
results=[ iztrans(X_z,i) for i in nparange(-N,N+1)]

fig = figure()
ax1 = fig.add_subplot(111)
ax1.set_ylabel('Valor',fontsize=15)
ax1.set_xlabel('N',fontsize=15)
ax1.set_title(title,fontsize=18)
stem(nparange(-N,N+1),results, use_line_collection=True,linefmt=None,
     markerfmt=None, basefmt=None)
show()
```

Listing 21: Simétrico da fórmula 2. Código em iztrans_7.Py

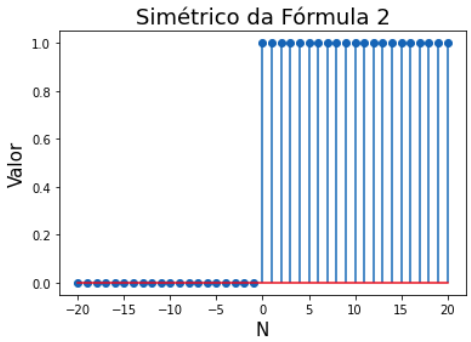

Figura 15: Resultado do código iztrans_7.Py

[8]

```
print(ztrans(
    -u(-n-1)
  )
)
```

Listing 22: Aplicação de ztrans. Código em iztrans_8.Py

```
-Piecewise((z/(-z + 1), Abs(z) < 1), (Sum(z**(-n), (n, -oo, -1)), True))
```

Figura 16: Resultado do código iztrans_8.Py

[9]

```
title="""Simétrico da Fórmula 3"""

N=20

X=Piecewise((z/(z - 1), True), (-Sum(z**(-n), (n, -oo, -1)), True))
m=0.01


results=[ iztrans(X,i,m) for i in nparange(-N,N+1)]

fig = figure()
ax1 = fig.add_subplot(111)
ax1.set_ylabel('Valor',fontsize=15)
ax1.set_xlabel('N',fontsize=15)
ax1.set_title(title,fontsize=18)
stem(nparange(-N,N+1),results, use_line_collection=True,linefmt=None, markerfmt=None,
    basefmt=None)
show()
```

Listing 23: Simétrico da fórmula 3. Código em iztrans_9.Py

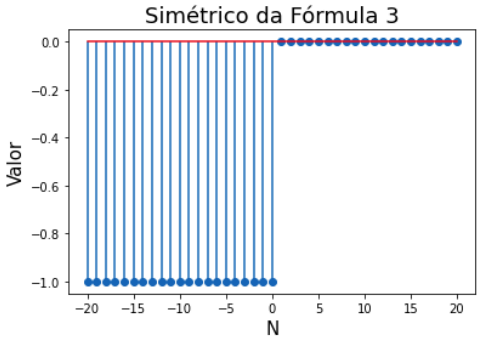

Figura 17: Resultado do código iztrans_9.Py

[10]

```
print(ztrans(
    DiracDelta(n-5)
  )
)
```

Listing 24: Aplicação de ztrans. Código em iztrans_10.Py

$$z^{**}(-5)$$

Figura 18: Resultado do código iztrans_10.Py

[11]

```
title="""Simétrico da Fórmula 4"""

N=20

X=z**(-5)

m=1.0


results=[ iztrans(X,i,m) for i in nparange(-N,N+1)]

fig = figure()
ax1 = fig.add_subplot(111)
ax1.set_ylabel('Valor',fontsize=15)
ax1.set_xlabel('N',fontsize=15)
ax1.set_title(title,fontsize=18)
stem(nparange(-N,N+1),results, use_line_collection=True,linefmt=None, markerfmt=None,
    basefmt=None)
show()
```

Listing 25: Simétrico da fórmula 4. Código em iztrans_11.Py

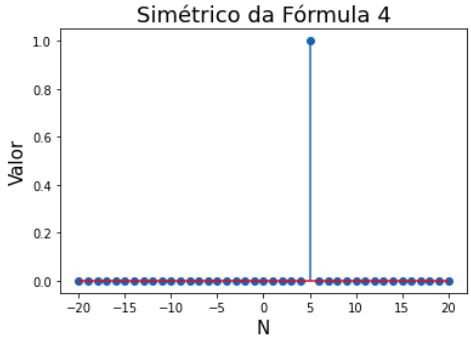

Figura 19: Resultado do código iztrans_11.Py

[12]

```
print(ztrans(
    a**n*u(n)
  )
)
```

Listing 26: Aplicação de ztrans. Código em iztrans_12.Py

```
Piecewise((1/(-a/z + 1), Abs(a/z) < 1), (Sum(a**n*z**(-n), (n, 0, oo)), True))
```

Figura 20: Resultado do código iztrans_12.Py

[13]

```
title="""Simétrico da Fórmula 5"""

N=6

X=Piecewise((z/(-a+z), True), (Sum(a**n*z**(-n), (n, 0, oo)), True))
X=X.subs(a,2)
# a deve ser substituído por um INTEIRO !

m=1.0


results=[ iztrans(X,i,m) for i in nparange(-N,N+1)]

fig = figure()
ax1 = fig.add_subplot(111)
ax1.set_ylabel('Valor',fontsize=15)
ax1.set_xlabel('N',fontsize=15)
ax1.set_title(title,fontsize=18)
stem(nparange(-N,N+1),results, use_line_collection=True,linefmt=None, markerfmt=None,
    basefmt=None)
show()
```

Listing 27: Simétrico da fórmula 5. Código em iztrans_13.Py

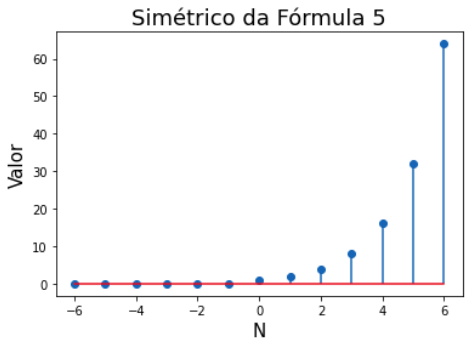

Figura 21: Resultado do código iztrans_13.Py

OBS: A limitação é que $a \in \mathbf{Z}$.

[14]

```
print(ztrans(
    -a**n*u(-n-1)
  )
)
```

Listing 28: Aplicação de ztrans. Código em iztrans_14.Py

```
-Piecewise((z/(a*(1 - z/a)), Abs(z/a) < 1), (Sum(a**n*z**(-n), (n, -oo, -1)), True))
```

Figura 22: Resultado do código iztrans_14.Py

[15]

```
title="""Simétrico da Fórmula 6"""

N=6

X=Piecewise((z/(-a + z), True), (-Sum(a**n*z**(-n), (n, -oo, -1)), True))
X=X.subs(a,2)
# a deve ser substituído por um INTEIRO !

#######
m=0.999
#######


results=[ iztrans(X,i,m) for i in nparange(-N,N+1)]

fig = figure()
ax1 = fig.add_subplot(111)
ax1.set_ylabel('Valor',fontsize=15)
ax1.set_xlabel('N',fontsize=15)
ax1.set_title(title,fontsize=18)
stem(nparange(-N,N+1),results, use_line_collection=True,linefmt=None,
     markerfmt=None, basefmt=None)
show()
```

Listing 29: Simétrico da fórmula 6. Código em iztrans_15.Py

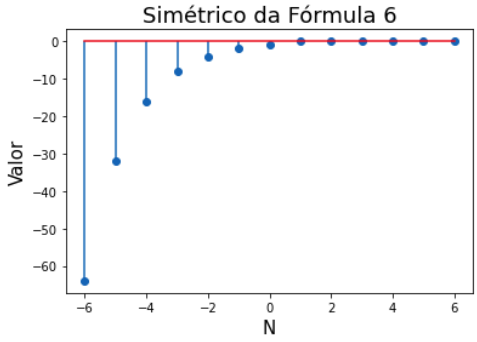

Figura 23: Resultado do código iztrans_15.Py

[16]

```
print(ztrans(
    n*a**n*u(n)
  )
)
```

Listing 30: Aplicação de ztrans. Código em iztrans_16.Py

```
Piecewise((a/(z*(-a/z + 1)**2), Abs(a/z) < 1), (Sum(a**n*n*z**(-n), (n, 0, oo)), True))
```

Figura 24: Resultado do código iztrans_16.Py

[17]

```
title="""Simétrico da Fórmula 7"""

N=6

X=Piecewise((a*z/(a - z)**2, True), (Sum(a**n*n*z**(-n), (n, 0, oo)), True))
X=X.subs(a,2)
# a deve ser substituído por um INTEIRO !

#######
m=39
#######


results=[ iztrans(X,i,m) for i in nparange(-N,N+1)]

fig = figure()
ax1 = fig.add_subplot(111)
ax1.set_ylabel('Valor',fontsize=15)
ax1.set_xlabel('N',fontsize=15)
ax1.set_title(title,fontsize=18)
stem(nparange(-N,N+1),results, use_line_collection=True,linefmt=None,
     markerfmt=None, basefmt=None)
show()
```

Listing 31: Simétrico da fórmula 7. Código em iztrans_17.Py

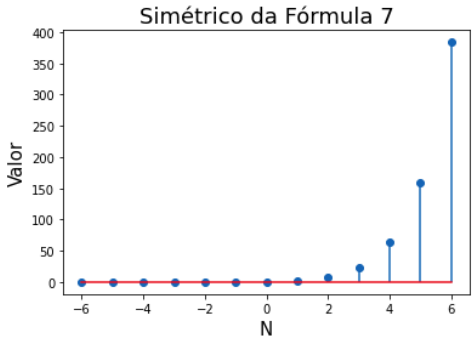

Figura 25: Resultado do código iztrans_17.Py

[18]

```
print(ztrans(
    -n*a**n*u(-n-1)
  )
)
```

Listing 32: Aplicação de ztrans. Código em iztrans_18.Py

```
-Piecewise((-z/(a*(1 - z/a)**2), Abs(z/a) < 1), (Sum(a**n*n*z**(-n), (n, -oo, -1)), True))
```

Figura 26: Resultado do código iztrans_18.Py

[19]

```
title="""Simétrico da Fórmula 8"""

N=6

X=Piecewise((a*z/(a**2 - 2*a*z + z**2), True), (-Sum(a**n*n*z**(-n), (n, -oo, -1)),
    True))
X=X.subs(a,2)
# a deve ser substituído por um INTEIRO !

#######
m=0.001
#######


results=[ iztrans(X,i,m) for i in nparange(-N,N+1)]

fig = figure()
ax1 = fig.add_subplot(111)
ax1.set_ylabel('Valor',fontsize=15)
ax1.set_xlabel('N',fontsize=15)
ax1.set_title(title,fontsize=18)
stem(nparange(-N,N+1),results, use_line_collection=True,linefmt=None,
     markerfmt=None, basefmt=None)
show()
```

Listing 33: Simétrico da fórmula 8. Código em iztrans_19.Py

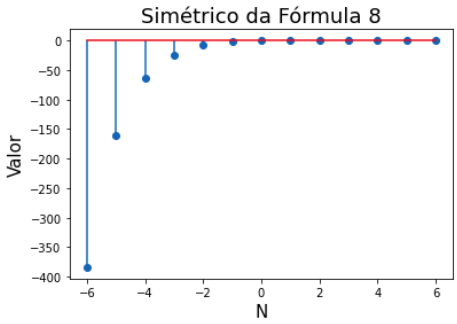

Figura 27: Resultado do código iztrans_19.Py

A priori, as fórmulas que correspondem aos números desde 9 até 12 poderiam ser obtidas a partir do simétrico da fórmula 5. Contudo, isso só funciona por meio do construído aqui para $a \in \mathbf{Z}$.