Introducing RSESS
An Open-Source Enumerative Sphere Shaping Implementation Coded in Rust

Frederik Ritter, Andrej Rode, and Laurent Schmalen
Why Probabilistic Shaping?

- Consistent growth of internet usage drives demand for higher bandwidth on fiber optical networks [ITU22]

Why Probabilistic Shaping?

- Consistent growth of internet usage drives demand for higher bandwidth on fiber optical networks [ITU22]
- Increase capacity of existing infrastructure

Why Probabilistic Shaping?

- Consistent growth of internet usage drives demand for higher bandwidth on fiber optical networks [ITU22]
- Increase capacity of existing infrastructure
- **Probabilistic shaping** achieves up to 1.53 dB shaping gain over uniformly distributed channel input symbols [FW89, KP93]

---


Why Probabilistic Shaping?

- Consistent growth of internet usage drives demand for higher bandwidth on fiber optical networks [ITU22]
- Increase capacity of existing infrastructure
- **Probabilistic shaping** achieves up to 1.53 dB shaping gain over uniformly distributed channel input symbols [FW89,KP93]
- **Our contribution:** A practical implementation of a probabilistic shaping algorithm for research and development


What are the Challenges of Probabilistic Shaping?

- Continuous channel input is not feasible with current systems

---

Uniform Data Bits Shaper Channel Deshaper

Uniform Data Bits Shaped Modulation Symbols Shaped Modulation Symbols


Ritter, Rode, Schmalen: Introducing RSESS Communications Engineering Lab (CEL)
What are the Challenges of Probabilistic Shaping?

- Continuous channel input is not feasible with current systems
- Adjust probabilities of discrete modulation symbols

Uniform Data Bits Shaper Channel Deshaper Uniform Data Bits Shaped Modulation Symbols Shaped Modulation Symbols

What are the Challenges of Probabilistic Shaping?

- Continuous channel input is not feasible with current systems
- Adjust probabilities of discrete modulation symbols
- Data bits follow a uniform distribution
What are the Challenges of Probabilistic Shaping?

- Continuous channel input is not feasible with current systems
- Adjust probabilities of discrete modulation symbols
- Data bits follow a uniform distribution

Incorporate forward error correction (FEC) to protect transmission

Probabilistic amplitude shaping (PAS) architecture

What are the Challenges of Probabilistic Shaping?

- Continuous channel input is not feasible with current systems
- Adjust probabilities of discrete modulation symbols
- Data bits follow a uniform distribution
- Incorporate forward error correction (FEC) to protect transmission

![Diagram showing the process of probabilistic shaping: Uniform Data Bits are input to the Shaper, which shapes the Modulation Symbols. These symbols are transmitted through the Channel, which also shapes them. The Deshaper then converts the shaped symbols back into Uniform Data Bits.]
What are the Challenges of Probabilistic Shaping?

- Continuous channel input is not feasible with current systems
- Adjust probabilities of discrete modulation symbols
- Data bits follow a uniform distribution
- Incorporate forward error correction (FEC) to protect transmission
- Probabilistic amplitude shaping (PAS) architecture [BS15] transmits parity information in the signs

![Diagram of probabilistic shaping process]

What are the Challenges of Probabilistic Shaping?

- Continuous channel input is not feasible with current systems
- Adjust probabilities of discrete modulation symbols
- Data bits follow a uniform distribution
- Incorporate forward error correction (FEC) to protect transmission
  - Probabilistic amplitude shaping (PAS) architecture [BS15] transmits parity information in the signs
  - Shaping only necessary for the amplitudes

![Diagram of the probabilistic shaping process]

How ESS Works

- Multiple probabilistic shaping algorithms exist:
  - Constant composition distribution matching [SB16]


How ESS Works

- Multiple probabilistic shaping algorithms exist:
  - Constant composition distribution matching [SB16]
  - Laroia’s algorithms in [LF94]

---


How ESS Works

- Multiple probabilistic shaping algorithms exist:
  - Constant composition distribution matching [SB16]
  - Laroia’s algorithms in [LF94]
- Enumerative sphere shaping (ESS) algorithm published in [WW93]


How ESS Works

- Multiple probabilistic shaping algorithms exist:
  - Constant composition distribution matching [SB16]
  - Laroia’s algorithms in [LF94]

- Enumerative sphere shaping (ESS) algorithm published in [WW93]

- Works on sequences of bits / amplitudes

---


How ESS Works

- Multiple probabilistic shaping algorithms exist:
  - Constant composition distribution matching [SB16]
  - Laroia’s algorithms in [LF94]
- Enumerative sphere shaping (ESS) algorithm published in [WW93]
- Works on sequences of bits / amplitudes

**Idea:** Reach AWGN capacity by using an energy efficient data representation

---


Multiple probabilistic shaping algorithms exist:
- Constant composition distribution matching [SB16]
- Laroia’s algorithms in [LF94]

**Enumerative sphere shaping (ESS) algorithm** published in [WW93]

Works on sequences of bits / amplitudes

**Idea:** Reach AWGN capacity by using an energy efficient data representation

Used amplitude sequences have minimal average energy

---


How ESS Works

- Multiple probabilistic shaping algorithms exist:
  - Constant composition distribution matching [SB16]
  - Laroia’s algorithms in [LF94]

- Enumerative sphere shaping (ESS) algorithm published in [WW93]

- Works on sequences of bits / amplitudes

- **Idea:** Reach AWGN capacity by using an energy efficient data representation

- Used amplitude sequences have minimal average energy

- **Relevant parameters:** Energy threshold $E_{\text{max}}$, amplitude sequence length $N$, and modulation order $M$

---


Amplitude sequence is a path through the trellis

Each transition is an amplitude $\in A = \{1, 3, 5, 7\}$

Each node represents a number of amplitudes $n$ and their total energy $e$.

Size of the trellis limits sequence energy

Value of a node $T^n_e$ is the number of possible continuations.
Amplitude sequence is a path through the trellis.

$T_{ne}^e$ is the number of possible continuations.
ESS Trellis Construction

- Amplitude sequence is a path through the trellis
- Each transition is an amplitude $\in A = \{1, 3, 5, 7\}$
ESS Trellis Construction

- Amplitude sequence is a path through the trellis
- Each transition is an amplitude \( \in A = \{1, 3, 5, 7\} \)
- Each node represents a number of amplitudes \( n \) and their total energy \( e \)
ESS Trellis Construction

- Amplitude sequence is a path through the trellis
- Each transition is an amplitude $\in A = \{1, 3, 5, 7\}$
- Each node represents a number of amplitudes $n$ and their total energy $e$
ESS Trellis Construction

- Amplitude sequence is a path through the trellis
- Each transition is an amplitude $\in A = \{1, 3, 5, 7\}$
- Each node represents a number of amplitudes $n$ and their total energy $e$
- Size of the trellis limits sequence energy
ESS Trellis Construction

- Amplitude sequence is a path through the trellis
- Each transition is an amplitude $\epsilon \in A = \{1, 3, 5, 7\}$
- Each node represents a number of amplitudes $n$ and their total energy $e$
- Size of the trellis limits sequence energy
- Value of a node $T_n^e$ is the number of possible continuations
ESS Trellis Construction

- Amplitude sequence is a path through the trellis
- Each transition is an amplitude $\in A = \{1, 3, 5, 7\}$
- Each node represents a number of amplitudes $n$ and their total energy $e$
- Size of the trellis limits sequence energy
- Value of a node $T_n^e$ is the number of possible continuations
ESS Trellis Construction

- Amplitude sequence is a path through the trellis.
- Each transition is an amplitude $\in A = \{1, 3, 5, 7\}$.
- Each node represents a number of amplitudes $n$ and their total energy $e$.
- Size of the trellis limits sequence energy.
- Value of a node $T^n_e$ is the number of possible continuations.
ESS Trellis Construction

- Amplitude sequence is a path through the trellis
- Each transition is an amplitude $\in A = \{1, 3, 5, 7\}$
- Each node represents a number of amplitudes $n$ and their total energy $e$
- Size of the trellis limits sequence energy
- Value of a node $T^n_e$ is the number of possible continuations
ESS Trellis Construction

- Amplitude sequence is a path through the trellis
- Each transition is an amplitude $\in A = \{1, 3, 5, 7\}$
- Each node represents a number of amplitudes $n$ and their total energy $e$
- Size of the trellis limits sequence energy
- Value of a node $T^n_e$ is the number of possible continuations
Amplitude sequence is a path through the trellis

Each transition is an amplitude $\in A = \{1, 3, 5, 7\}$

Each node represents a number of amplitudes $n$ and their total energy $e$

Size of the trellis limits sequence energy

Value of a node $T^n_e$ is the number of possible continuations

---

**ESS Trellis Construction**

![Trellis Diagram]

- $T^n_e$ represents the energy of the $n$th node.
- The diagram shows transitions from one node to another, representing amplitude sequences.
- Each node is labeled with its total energy and node number.
- The trellis diagram illustrates the structure of the sequence energy with transitions labeled by amplitudes from the set $A = \{1, 3, 5, 7\}$.
What is RSESS?

- **RSESS**: Rust library implementing ESS (and Optimum ESS [cc22])


github.com/kit-cel/rsess
What is RSESS?

- **RSESS**: Rust library implementing **ESS** (and Optimum ESS [cc22])
- Implements encoding, decoding as well as multiple utility functions

What is RSESS?

- **RSESS**: Rust library implementing ESS (and Optimum ESS [cc22])
- Implements encoding, decoding as well as multiple utility functions
- **PyRSESS** is a Python binding for the full scope of RSESS

What is RSESS?

- **RSESS**: Rust library implementing **ESS** (and Optimum ESS [cc22])
- Implements encoding, decoding as well as multiple utility functions
- **PyRSESS** is a Python binding for the full scope of RSESS
- Published to crates.io, pypi.org, and github.com

A Simple Example

from pyrsess import ESS

ess = ESS(28, 4, 8)  # ESS instance with $E_{\text{max}} = 28$, $N = 4$, and 8-ASK

data = [1, 1, 0, 1]  # 1101 (binary) == index 13

sequence = ess.encode(data)
print(sequence)  # [3 1 3 1]

data_decoded = ess.decode(sequence)
print(data_decoded)  # [1 1 0 1]
from pyrsess import ESS

ess = ESS(28, 4, 8)  # ESS instance with \( E_{\text{max}} = 28 \), \( N = 4 \), and 8-ASK

data = [1, 1, 0, 1]  # 1101 (binary) == index 13

sequence = ess.encode(data)
print(sequence)  # [3 1 3 1]

data_decoded = ess.decode(sequence)
print(data_decoded)  # [1 1 0 1]
A Simple Example

```python
from pyrsess import ESS

ess = ESS(28, 4, 8)  # ESS instance with E_max = 28, N = 4, and 8-ASK

data = [1, 1, 0, 1]  # 1101 (binary) == index 13

sequence = ess.encode(data)
print(sequence)  # [3 1 3 1]

data_decoded = ess.decode(sequence)
print(data_decoded)  # [1 1 0 1]
```
from pyrsess import ESS

ess = ESS(28, 4, 8)  # ESS instance with $E_{\text{max}} = 28$, $N = 4$, and 8-ASK

data = [1, 1, 0, 1]  # 1101 (binary) == index 13

sequence = ess.encode(data)
print(sequence)
from pyrsess import ESS

ess = ESS(28, 4, 8)  # ESS instance with E_max = 28, N = 4, and 8-ASK

data = [1, 1, 0, 1]  # 1101 (binary) == index 13

sequence = ess.encode(data)
print(sequence)

[3 1 3 1]
A Simple Example

```python
from pyrsess import ESS

ess = ESS(28, 4, 8)  # ESS instance with E_max = 28, N = 4, and 8-ASK

data = [1, 1, 0, 1]  # 1101 (binary) == index 13

sequence = ess.encode(data)
print(sequence)

[3 1 3 1]

data_decoded = ess.decode(sequence)
print(data_decoded)
```

`[1 1 0 1]`
from pyrsess import ESS

ess = ESS(28, 4, 8)  # ESS instance with E_max = 28, N = 4, and 8-ASK

data = [1, 1, 0, 1]  # 1101 (binary) == index 13

sequence = ess.encode(data)
print(sequence)

[3 1 3 1]

data_decoded = ess.decode(sequence)
print(data_decoded)

[1 1 0 1]
RSESS is Fast

- Measured time for 10000 encodings / decodings and 8-ASK

![Graph showing run time (µs) vs. Amplitude sequence length for Encode RSESS, Decode RSESS, Encode PyRSESS, Decode PyRSESS]
RSESS is Fast

- Measured time for 10000 encodings / decodings and 8-ASK
- Shaping rate (number of bits per amplitude) held constant at 1.5
RSESS is Fast

- Measured time for 10000 encodings / decodings and 8-ASK
- Shaping rate (number of bits per amplitude) held constant at 1.5
- Tested on a 4-core Intel Core i7-7700 (2017)
RSESS is Fast

- Measured time for 10000 encodings / decodings and 8-ASK
- Shaping rate (number of bits per amplitude) held constant at 1.5
- Tested on a 4-core Intel Core i7-7700 (2017)

![Graph showing run time and memory usage vs amplitude sequence length]
RSESS Validation

- Results using RSESS match up with literature
RSESS Validation

- Results using RSESS match up with literature

Results on AWGN channel with RSESS

Reference results in [CC22]

Conclusion

- Non-uniform transmission can increase the possible data rate
Conclusion

- Non-uniform transmission can increase the possible data rate
- An open-source implementation of ESS is now available

github.com/kit-cel/rsess
Conclusion

- Non-uniform transmission can increase the possible data rate
- An open-source implementation of ESS is now available
- RSESS is fast thanks to Rust

github.com/kit-cel/rsess

Questions?

This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 101001899).
Conclusion

- Non-uniform transmission can increase the possible data rate
- An open-source implementation of ESS is now available
- RSESS is fast thanks to Rust
- RSESS results match literature results

github.com/kit-cel/rsess

Questions?

This work has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No. 101001899).
Conclusion

- Non-uniform transmission can increase the possible data rate
- An open-source implementation of ESS is now available
- RSESS is fast thanks to Rust
- RSESS results match literature results

Questions?

This work has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No. 101001899).