COMP4121 Advanced Algorithms

Shuiqiao Yang

School of Computer Science and Engineering
University of New South Wales Sydney
More fixed point algorithms:
Transmit Power Control In Cellular Networks
Introduction

- Assume you are at a party, talking to someone.

- You can also hear people around you chatting with each other.

- You can also hear the noisy traffic outside, so you instinctively start talking a bit louder.

- However, your conversation is noise to other people, so they also increase their volume too.

- You start talking even louder, and pretty soon everyone is almost shouting.

- Something like that can potentially happen in cellular networks; separation between channels on which users communicate is imperfect, so there is always interference between the users.
Introduction

• Assume you are at a party, talking to someone.

• You can also hear people around you chatting with each other.

• You can also hear the noisy traffic outside, so you instinctively start talking a bit louder.

• However, your conversation is noise to other people, so they also increase their volume too.

• You start talking even louder, and pretty soon everyone is almost shouting.

• Something like that can potentially happen in cellular networks; separation between channels on which users communicate is imperfect, so there is always interference between the users.
Assume you are at a party, talking to someone.

You can also hear people around you chatting with each other.

You can also hear the noisy traffic outside, so you instinctively start talking a bit louder.

However, your conversation is noise to other people, so they also increase their volume too.

You start talking even louder, and pretty soon everyone is almost shouting.

Something like that can potentially happen in cellular networks; separation between channels on which users communicate is imperfect, so there is always interference between the users.
Introduction

- Assume you are at a party, talking to someone.

- You can also hear people around you chatting with each other.

- You can also hear the noisy traffic outside, so you instinctively start talking a bit louder.

- However, your conversation is noise to other people, so they also increase their volume too.

- You start talking even louder, and pretty soon everyone is almost shouting.

- Something like that can potentially happen in cellular networks; separation between channels on which users communicate is imperfect, so there is always interference between the users.
Assume you are at a party, talking to someone.

You can also hear people around you chatting with each other.

You can also hear the noisy traffic outside, so you instinctively start talking a bit louder.

However, your conversation is noise to other people, so they also increase their volume too.

You start talking even louder, and pretty soon everyone is almost shouting.

Something like that can potentially happen in cellular networks; separation between channels on which users communicate is imperfect, so there is always interference between the users.
Assume you are at a party, talking to someone.

You can also hear people around you chatting with each other.

You can also hear the noisy traffic outside, so you instinctively start talking a bit louder.

However, your conversation is noise to other people, so they also increase their volume too.

You start talking even louder, and pretty soon everyone is almost shouting.

Something like that can potentially happen in cellular networks; separation between channels on which users communicate is imperfect, so there is always interference between the users.
The gain $G_{ij}$ of a communication channel $C_{ij}$ between a transmitter $T_j$ and a receiver $R_i$ is the quantity such that if $p_j$ is the transmit power of transmitter $T_j$, then the power of the signal received by $R_i$ is $G_{ij}p_j$.

Clearly, $G_{ij}$ is (much) smaller than 1, so $G_{ij}$ would be better called channel attenuation, but engineers use term channel gain.

Let us say there are $n$ pairs of transmitters $T_i$ and receivers $R_i$.

Between a transmitter $T_i$ and receiver $R_i$ the direct channel of communication $C_{ii}$ has a gain $G_{ii}$.

The spurious channels $C_{ij}$ between a transmitter $T_j$ and a receiver $R_i$, $i \neq j$, have gains $G_{ij}$.

Gain $G_{ii}$ of a direct channel $C_{ii}$ is significantly larger than the gains of spurious channels $C_{ij}$. 
• The gain $G_{ij}$ of a communication channel $C_{ij}$ between a transmitter $T_j$ and a receiver $R_i$ is the quantity such that if $p_j$ is the transmit power of transmitter $T_j$, then the power of the signal received by $R_i$ is $G_{ij}p_j$.

• Clearly, $G_{ij}$ is (much) smaller than 1, so $G_{ij}$ would be better called channel attenuation, but engineers use term *channel gain*.

• Let us say there are $n$ pairs of transmitters $T_i$ and receivers $R_i$.

• Between a transmitter $T_i$ and receiver $R_i$ the direct channel of communication $C_{ii}$ has a gain $G_{ii}$.

• The spurious channels $C_{ij}$ between a transmitter $T_j$ and a receiver $R_i$, $i \neq j$, have gains $G_{ij}$.

• Gain $G_{ii}$ of a direct channel $C_{ii}$ is significantly larger than the gains of spurious channels $C_{ij}$.
The gain $G_{ij}$ of a communication channel $C_{ij}$ between a transmitter $T_j$ and a receiver $R_i$ is the quantity such that if $p_j$ is the transmit power of transmitter $T_j$, then the power of the signal received by $R_i$ is $G_{ij} p_j$.

Clearly, $G_{ij}$ is (much) smaller than 1, so $G_{ij}$ would be better called channel attenuation, but engineers use term channel gain.

Let us say there are $n$ pairs of transmitters $T_i$ and receivers $R_i$.

Between a transmitter $T_i$ and receiver $R_i$ the direct channel of communication $C_{ii}$ has a gain $G_{ii}$.

The spurious channels $C_{ij}$ between a transmitter $T_j$ and a receiver $R_i$, $i \neq j$, have gains $G_{ij}$.

Gain $G_{ii}$ of a direct channel $C_{ii}$ is significantly larger than the gains of spurious channels $C_{ij}$.
The gain $G_{ij}$ of a communication channel $C_{ij}$ between a transmitter $T_j$ and a receiver $R_i$ is the quantity such that if $p_j$ is the transmit power of transmitter $T_j$, then the power of the signal received by $R_i$ is $G_{ij}p_j$.

Clearly, $G_{ij}$ is (much) smaller than 1, so $G_{ij}$ would be better called channel attenuation, but engineers use term channel gain.

Let us say there are $n$ pairs of transmitters $T_i$ and receivers $R_i$.

Between a transmitter $T_i$ and receiver $R_i$ the direct channel of communication $C_{ii}$ has a gain $G_{ii}$.

The spurious channels $C_{ij}$ between a transmitter $T_j$ and a receiver $R_i$, $i \neq j$, have gains $G_{ij}$.

Gain $G_{ii}$ of a direct channel $C_{ii}$ is significantly larger than the gains of spurious channels $C_{ij}$. 
The gain $G_{ij}$ of a communication channel $C_{ij}$ between a transmitter $T_j$ and a receiver $R_i$ is the quantity such that if $p_j$ is the transmit power of transmitter $T_j$, then the power of the signal received by $R_i$ is $G_{ij}p_j$.

Clearly, $G_{ij}$ is (much) smaller than 1, so $G_{ij}$ would be better called channel attenuation, but engineers use term channel gain.

Let us say there are $n$ pairs of transmitters $T_i$ and receivers $R_i$.

Between a transmitter $T_i$ and receiver $R_i$ the direct channel of communication $C_{ii}$ has a gain $G_{ii}$.

The spurious channels $C_{ij}$ between a transmitter $T_j$ and a receiver $R_i$, $i \neq j$, have gains $G_{ij}$.

Gain $G_{ii}$ of a direct channel $C_{ii}$ is significantly larger than the gains of spurious channels $C_{ij}$.
The gain $G_{ij}$ of a communication channel $C_{ij}$ between a transmitter $T_j$ and a receiver $R_i$ is the quantity such that if $p_j$ is the transmit power of transmitter $T_j$, then the power of the signal received by $R_i$ is $G_{ij}p_j$.

Clearly, $G_{ij}$ is (much) smaller than 1, so $G_{ij}$ would be better called channel attenuation, but engineers use term channel gain.

Let us say there are $n$ pairs of transmitters $T_i$ and receivers $R_i$.

Between a transmitter $T_i$ and receiver $R_i$ the direct channel of communication $C_{ii}$ has a gain $G_{ii}$.

The spurious channels $C_{ij}$ between a transmitter $T_j$ and a receiver $R_i$, $i \neq j$, have gains $G_{ij}$.

Gain $G_{ii}$ of a direct channel $C_{ii}$ is significantly larger than the gains of spurious channels $C_{ij}$. 


**Figure:** Several transmitters $T_i$ trying to communicate with their corresponding receivers $R_i$.

- Gains of direct, intended communication channels: $G_{11}, G_{22}, G_{33}$.
- Gains of all other, unintended interference communication channels, whose existence is a consequence of imperfect separation between channels: $G_{ij}, i \neq j$. 
Figure: One possible structure of a cellular network with 7 (almost) disjoint frequency bands.
The signal to interference ratio $SIR_i$ at a receiver $R_i$ is given by

$$SIR_i = \frac{G_{ii}p_i}{\sum_{j:j\neq i} G_{ij}p_j + \eta_i}$$

- $\eta_i$ is the noise received by receiver $i$ coming from the environment, such as spark plugs of cars, electric machines etc., plus the noise introduced by the receiver’s own circuitry and algorithms.

- $SIR_i$ determines the capacity of the channel $C_{ii}$. 
The signal to interference ratio $SIR_i$ at a receiver $R_i$ is given by

$$SIR_i = \frac{G_{ii}p_i}{\sum_{j:j \neq i} G_{ij}p_j + \eta_i}$$

$\eta_i$ is the noise received by receiver $i$ coming from the environment, such as spark plugs of cars, electric machines etc., plus the noise introduced by the receiver’s own circuitry and algorithms.

$SIR_i$ determines the capacity of the channel $C_{ii}$. 
The signal to interference ratio $SIR_i$ at a receiver $R_i$ is given by

$$SIR_i = \frac{G_{ii}p_i}{\sum_{j:j\neq i} G_{ij}p_j + \eta_i}$$

- $\eta_i$ is the noise received by receiver $i$ coming from the environment, such as spark plugs of cars, electric machines etc., plus the noise introduced by the receiver’s own circuitry and algorithms.

- $SIR_i$ determines the capacity of the channel $C_{ii}$. 
Each pair of a transmitter $T_i$ and a receiver $R_i$ needs certain channel capacity to carry information.

In order to achieve this, it needs the value of $SIR_i$ to be at least $\gamma_i$.

The task of a transmit power control algorithm is to determine the powers $p_i$, $1 \leq i \leq n$, which:

1. provide each receiver with $SIR_i$ of at least $\gamma_i$;
2. the total power of all transmitters $\sum_{1 \leq j \leq n} p_j$ is as small as possible.

We will later see how $\gamma_i$ are determined in practice.

At the moment we are interested in the following minimisation problem:

\[
\text{minimise} \quad \sum_{j=1}^{n} p_j \\
\text{subject to constraints} \quad \frac{G_{ii}p_i}{\sum_{j:j \neq i} G_{ij}p_j + \eta_i} \geq \gamma_i, \quad 1 \leq i \leq n
\]
Each pair of a transmitter $T_i$ and a receiver $R_i$ needs certain channel capacity to carry information.

In order to achieve this, it needs the value of $SIR_i$ to be at least $\gamma_i$.

The task of a transmit power control algorithm is to determine the powers $p_i$, $1 \leq i \leq n$, which:

1. provide each receiver with $SIR_i$ of at least $\gamma_i$;
2. the total power of all transmitters $\sum_{1 \leq j \leq n} p_j$ is as small as possible.

We will later see how $\gamma_i$ are determined in practice.

At the moment we are interested in the following minimisation problem:

minimise $\sum_{j=1}^{n} p_j$

subject to constraints $\frac{G_{ii}p_i}{\sum_{j:j \neq i} G_{ij}p_j + \eta_i} \geq \gamma_i, \quad 1 \leq i \leq n$
• Each pair of a transmitter $T_i$ and a receiver $R_i$ needs certain channel capacity to carry information.

• In order to achieve this, it needs the value of $SIR_i$ to be at least $\gamma_i$.

• The task of a transmit power control algorithm is to determine the powers $p_i$, $1 \leq i \leq n$, which:
  1. provide each receiver with $SIR_i$ of at least $\gamma_i$;
  2. the total power of all transmitters $\sum_{1 \leq j \leq n} p_j$ is as small as possible.

• We will later see how $\gamma_i$ are determined in practice.

• At the moment we are interested in the following minimisation problem:

$$\text{minimise} \quad \sum_{j=1}^{n} p_j$$

subject to constraints

$$\frac{G_{ii}p_i}{\sum_{j:j \neq i} G_{ij}p_j + \eta_i} \geq \gamma_i, \quad 1 \leq i \leq n$$
Each pair of a transmitter $T_i$ and a receiver $R_i$ needs certain channel capacity to carry information.

In order to achieve this, it needs the value of $SIR_i$ to be at least $\gamma_i$.

The task of a transmit power control algorithm is to determine the powers $p_i$, $1 \leq i \leq n$, which:

1. provide each receiver with $SIR_i$ of at least $\gamma_i$;
2. the total power of all transmitters $\sum_{1 \leq j \leq n} p_j$ is as small as possible.

We will later see how $\gamma_i$ are determined in practice.

At the moment we are interested in the following minimisation problem:

$$\text{minimise} \quad \sum_{j=1}^{n} p_j$$

subject to constraints

$$\frac{G_{ii} p_i}{\sum_{j:j \neq i} G_{ij} p_j + \eta_i} \geq \gamma_i, \quad 1 \leq i \leq n$$
Each pair of a transmitter \( T_i \) and a receiver \( R_i \) needs certain channel capacity to carry information.

In order to achieve this, it needs the value of \( SIR_i \) to be at least \( \gamma_i \).

The task of a transmit power control algorithm is to determine the powers \( p_i \), \( 1 \leq i \leq n \), which:

1. provide each receiver with \( SIR_i \) of at least \( \gamma_i \);
2. the total power of all transmitters \( \sum_{1 \leq j \leq n} p_j \) is as small as possible.

We will later see how \( \gamma_i \) are determined in practice.

At the moment we are interested in the following minimisation problem:

\[
\text{minimise } \sum_{j=1}^{n} p_j \\
\text{subject to constraints } \frac{G_{ii}p_i}{\sum_{j:j \neq i} G_{ij}p_j + \eta_i} \geq \gamma_i, \quad 1 \leq i \leq n
\]
Each pair of a transmitter $T_i$ and a receiver $R_i$ needs certain channel capacity to carry information.

In order to achieve this, it needs the value of $SIR_i$ to be at least $\gamma_i$.

The task of a transmit power control algorithm is to determine the powers $p_i$, $1 \leq i \leq n$, which:

1. provide each receiver with $SIR_i$ of at least $\gamma_i$;
2. the total power of all transmitters $\sum_{1 \leq j \leq n} p_j$ is as small as possible.

We will later see how $\gamma_i$ are determined in practice.

At the moment we are interested in the following minimisation problem:

$$\text{minimise} \quad \sum_{j=1}^{n} p_j$$

subject to constraints

$$\frac{G_{ii}p_i}{\sum_{j:j \neq i} G_{ij}p_j + \eta_i} \geq \gamma_i, \quad 1 \leq i \leq n$$
• Each pair of a transmitter $T_i$ and a receiver $R_i$ needs certain channel capacity to carry information.

• In order to achieve this, it needs the value of $SIR_i$ to be at least $\gamma_i$.

• The task of a transmit power control algorithm is to determine the powers $p_i$, $1 \leq i \leq n$, which:
  1. provide each receiver with $SIR_i$ of at least $\gamma_i$;
  2. the total power of all transmitters $\sum_{1 \leq j \leq n} p_j$ is as small as possible.

• We will later see how $\gamma_i$ are determined in practice.

• At the moment we are interested in the following minimisation problem:

  \[
  \text{minimise} \quad \sum_{j=1}^{n} p_j \\
  \text{subject to constraints} \quad \frac{G_{ii}p_i}{\sum_{j: j \neq i} G_{ij}p_j + \eta_i} \geq \gamma_i, \quad 1 \leq i \leq n
  \]
This minimisation problem

\[
\text{minimise} \quad \sum_{j=1}^{n} p_j \\
\text{subject to constraints} \quad \frac{G_{ii}p_i}{\sum_{j:j \neq i} G_{ij}p_j + \eta_i} \geq \gamma_i, \quad 1 \leq i \leq n
\]

can be replaced by an equivalent problem linear in the unknown variables \( p_j \):

\[
\text{minimise} \quad \sum_{j=1}^{n} p_j \\
\text{subject to constraints} \quad G_{ii}p_i \geq \gamma_i \left( \sum_{j:j \neq i} G_{ij}p_j + \eta_i \right), \quad 1 \leq i \leq n \\
p_j > 0, \quad 1 \leq j \leq n
\]
Such a linear program:

\[
\text{minimise } \sum_{j=1}^{n} p_j
\]

subject to constraints

\[
G_{ii} p_i \geq \gamma_i \left( \sum_{j:j \neq i} G_{ij} p_j + \eta_i \right), \quad 1 \leq i \leq n
\]

\[
p_j > 0, \quad 1 \leq j \leq n
\]

has a simpler but equivalent formulation:

\[
\text{minimise } \sum_{j=1}^{n} p_j \tag{1}
\]

subject to constraints

\[
p_i - \gamma_i \sum_{j:j \neq i} \frac{G_{ij}}{G_{ii}} p_j \geq \frac{\gamma_i \eta_i}{G_{ii}}, \quad 1 \leq i \leq n \tag{2}
\]

\[
p_j > 0, \quad 1 \leq j \leq n \tag{3}
\]
A linear programming problem can be solved efficiently but this requires significant computational resources.

Moreover, all transmitters and receivers would have to exchange information about their corresponding $\gamma_i$ and $G_{ij}$ values.

Then they would have to solve a rather large optimisation problem, which would be inconvenient.

Ideally, each pair of a transmitter and a receiver should be able to make adjustments only from data available to them, and yet somehow achieve a globally optimal solution.

Remarkably, this is indeed possible by a very simple algorithm but with a non-trivial justification of optimality.
A linear programming problem can be solved efficiently but this requires significant computational resources.

Moreover, all transmitters and receivers would have to exchange information about their corresponding $\gamma_i$ and $G_{ij}$ values.

Then they would have to solve a rather large optimisation problem, which would be inconvenient.

Ideally, each pair of a transmitter and a receiver should be able to make adjustments only from data available to them, and yet somehow achieve a globally optimal solution.

Remarkably, this is indeed possible by a very simple algorithm but with a non-trivial justification of optimality.
A linear programming problem can be solved efficiently but this requires significant computational resources.

Moreover, all transmitters and receivers would have to exchange information about their corresponding $\gamma_i$ and $G_{ij}$ values.

Then they would have to solve a rather large optimisation problem, which would be inconvenient.

Ideally, each pair of a transmitter and a receiver should be able to make adjustments only from data available to them, and yet somehow achieve a globally optimal solution.

Remarkably, this is indeed possible by a very simple algorithm but with a non-trivial justification of optimality.
• A linear programming problem can be solved efficiently but this requires significant computational resources.

• Moreover, all transmitters and receivers would have to exchange information about their corresponding $\gamma_i$ and $G_{ij}$ values.

• Then they would have to solve a rather large optimisation problem, which would be inconvenient.

• Ideally, each pair of a transmitter and a receiver should be able to make adjustments only from data available to them, and yet somehow achieve a globally optimal solution.

• Remarkably, this is indeed possible by a very simple algorithm but with a non-trivial justification of optimality.
A linear programming problem can be solved efficiently but this requires significant computational resources.

Moreover, all transmitters and receivers would have to exchange information about their corresponding \( \gamma_i \) and \( G_{ij} \) values.

Then they would have to solve a rather large optimisation problem, which would be inconvenient.

Ideally, each pair of a transmitter and a receiver should be able to make adjustments only from data available to them, and yet somehow achieve a globally optimal solution.

Remarkably, this is indeed possible by a very simple algorithm but with a non-trivial justification of optimality.
• Each pair transmitter/receiver $T_i/P_i$ (actually in mobile networks they are both transceivers, because the communication is bilateral) perform a very simple operation.

• We assume that time is divided into equal length time slots of duration of order of a few milliseconds.

• If $SIR_i(t)$ was the signal to interference ratio at the receiver $R_i$ during a time slot $t$, then during time slot $t + 1$ the transmitter will transmit with power

$$p_i(t + 1) = \frac{\gamma_i}{SIR_i(t)} p_i(t)$$

• Thus, if actual $SIR_i(t)$ was higher than the target $\gamma_i$, the new transmit power will be correspondingly lower and vice versa.

• But why would such an algorithm produce a globally optimal solution which minimises $\sum_{k=1}^{n} p_k$?
Each pair transmitter/receiver $T_i/P_i$ (actually in mobile networks they are both transceivers, because the communication is bilateral) perform a very simple operation.

We assume that time is divided into equal length time slots of duration of order of a few milliseconds.

If $SIR_i(t)$ was the signal to interference ratio at the receiver $R_i$ during a time slot $t$, then during time slot $t+1$ the transmitter will transmit with power

$$p_i(t + 1) = \frac{\gamma_i}{SIR_i(t)} p_i(t)$$

Thus, if actual $SIR_i(t)$ was higher than the target $\gamma_i$, the new transmit power will be correspondingly lower and vice versa.

But why would such an algorithm produce a globally optimal solution which minimises $\sum_{k=1}^{n} p_k$?
Each pair transmitter/receiver $T_i/P_i$ (actually in mobile networks they are both transceivers, because the communication is bilateral) perform a very simple operation.

We assume that time is divided into equal length time slots of duration of order of a few milliseconds.

If $SIR_i(t)$ was the signal to interference ratio at the receiver $R_i$ during a time slot $t$, then during time slot $t + 1$ the transmitter will transmit with power

$$p_i(t + 1) = \frac{\gamma_i}{SIR_i(t)} p_i(t)$$

Thus, if actual $SIR_i(t)$ was higher than the target $\gamma_i$, the new transmit power will be correspondingly lower and vice versa.

But why would such an algorithm produce a globally optimal solution which minimises $\sum_{k=1}^{n} p_k$?
Each pair transmitter/receiver $T_i/P_i$ (actually in mobile networks they are both transceivers, because the communication is bilateral) perform a very simple operation.

We assume that time is divided into equal length time slots of duration of order of a few milliseconds.

If $SIR_i(t)$ was the signal to interference ratio at the receiver $R_i$ during a time slot $t$, then during time slot $t + 1$ the transmitter will transmit with power

$$p_i(t + 1) = \frac{\gamma_i}{SIR_i(t)} p_i(t)$$

Thus, if actual $SIR_i(t)$ was higher than the target $\gamma_i$, the new transmit power will be correspondingly lower and vice versa.

But why would such an algorithm produce a globally optimal solution which minimises $\sum_{k=1}^{n} p_k$?
Each pair transmitter/receiver $T_i/P_i$ (actually in mobile networks they are both transceivers, because the communication is bilateral) perform a very simple operation.

We assume that time is divided into equal length time slots of duration of order of a few milliseconds.

If $SIR_i(t)$ was the signal to interference ratio at the receiver $R_i$ during a time slot $t$, then during time slot $t + 1$ the transmitter will transmit with power

$$p_i(t + 1) = \frac{\gamma_i}{SIR_i(t)} p_i(t)$$

Thus, if actual $SIR_i(t)$ was higher than the target $\gamma_i$, the new transmit power will be correspondingly lower and vice versa.

But why would such an algorithm produce a globally optimal solution which minimises $\sum_{k=1}^{n} p_k$?
We again transform the above minimisation problem into a matrix form.

Let
\[ \mathbf{1}^T = (1, 1, \ldots, 1); \quad \mathbf{v}^T = \left( \frac{\gamma_1 \eta_1}{G_{11}}, \ldots, \frac{\gamma_i \eta_i}{G_{ii}}, \ldots, \frac{\gamma_n \eta_n}{G_{nn}} \right) \]

Let also
\[
D = \begin{pmatrix}
\gamma_1 & 0 & \cdots & 0 & 0 \\
0 & \gamma_2 & \cdots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & \gamma_{n-1} & 0 \\
0 & 0 & \cdots & 0 & \gamma_n
\end{pmatrix} ;
\]
\[
F = \begin{pmatrix}
0 & \frac{G_{12}}{G_{11}} & \frac{G_{13}}{G_{11}} & \cdots & \frac{G_{1n}}{G_{11}} \\
\frac{G_{21}}{G_{11}} & 0 & \frac{G_{23}}{G_{22}} & \cdots & \frac{G_{2n}}{G_{22}} \\
\frac{G_{31}}{G_{22}} & \frac{G_{32}}{G_{22}} & 0 & \cdots & \frac{G_{3n}}{G_{32}} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\frac{G_{n1}}{G_{nn}} & \frac{G_{n2}}{G_{nn}} & \frac{G_{n3}}{G_{nn}} & \cdots & 0
\end{pmatrix} ;
\]

It is now easy to verify that our optimisation problem

\[
\begin{align*}
\text{minimise} & \quad \sum_{j=1}^n p_j \\
\text{subject to constraints} & \quad p_i - \gamma_i \sum_{j:j \neq i} \frac{G_{ij}}{G_{ii}} p_j \geq \frac{\gamma_i \eta_i}{G_{ii}}, \quad 1 \leq i \leq n \\
& \quad p_j > 0, \quad 1 \leq j \leq n
\end{align*}
\]

becomes:

\[
\begin{align*}
\text{minimise} & \quad \mathbf{1}^T \mathbf{p} \\
\text{subject to constraints} & \quad (I - DF) \mathbf{p} \geq \mathbf{v} \\
& \quad \mathbf{p} \geq 0
\end{align*}
\]
Here the inequalities \((I - DF)p \geq v\) and \(p \geq 0\) are defined coordinate-wise, i.e., if \(v, u \in \mathbb{R}^n\) are any two vectors, then \(v \geq u\) stands for \(v_i \geq u_i\) for all \(1 \leq i \leq n\).

When does such optimisation problem have feasible solutions (in the sense of a linear programming problem, i.e., solutions which satisfy the constraints)?

Intuitively, this should be possible if we are not demanding excessively large \(\gamma_i\)'s.

We now show that this is indeed the case when the spectral radius of matrix \(DF\) satisfies \(\rho(DF) < 1\).

Recall that the spectral radius \(\rho(X)\) of a matrix \(X\) is the largest absolute value of its eigenvalues.
Here the inequalities \((I - DF)p \geq v\) and \(p \geq 0\) are defined coordinate-wise, i.e., if \(v, u \in \mathbb{R}^n\) are any two vectors, then \(v \geq u\) stands for \(v_i \geq u_i\) for all \(1 \leq i \leq n\).

When does such optimisation problem have feasible solutions (in the sense of a linear programming problem, i.e., solutions which satisfy the constraints)?

Intuitively, this should be possible if we are not demanding excessively large \(\gamma_i's\).

We now show that this is indeed the case when the spectral radius of matrix \(DF\) satisfies \(\rho(DF) < 1\).

Recall that the spectral radius \(\rho(X)\) of a matrix \(X\) is the largest absolute value of its eigenvalues.
Here the inequalities \((I - DF)p \geq v\) and \(p \geq 0\) are defined coordinate-wise, i.e., if \(v, u \in \mathbb{R}^n\) are any two vectors, then \(v \geq u\) stands for \(v_i \geq u_i\) for all \(1 \leq i \leq n\).

When does such optimisation problem have feasible solutions (in the sense of a linear programming problem, i.e., solutions which satisfy the constraints)?

Intuitively, this should be possible if we are not demanding excessively large \(\gamma_i's\).

We now show that this is indeed the case when the spectral radius of matrix \(DF\) satisfies \(\rho(DF) < 1\).

Recall that the spectral radius \(\rho(X)\) of a matrix \(X\) is the largest absolute value of its eigenvalues.
Here the inequalities \((I - DF)p \geq v\) and \(p \geq 0\) are defined coordinate-wise, i.e., if \(v, u \in \mathbb{R}^n\) are any two vectors, then \(v \geq u\) stands for \(v_i \geq u_i\) for all \(1 \leq i \leq n\).

When does such optimisation problem have feasible solutions (in the sense of a linear programming problem, i.e., solutions which satisfy the constraints)?

Intuitively, this should be possible if we are not demanding excessively large \(\gamma_i\)'s.

We now show that this is indeed the case when the spectral radius of matrix \(DF\) satisfies \(\rho(DF) < 1\).

Recall that the spectral radius \(\rho(X)\) of a matrix \(X\) is the largest absolute value of its eigenvalues.
Here the inequalities \((I - DF)p \geq v\) and \(p \geq 0\) are defined coordinate-wise, i.e., if \(v, u \in \mathbb{R}^n\) are any two vectors, then \(v \geq u\) stands for \(v_i \geq u_i\) for all \(1 \leq i \leq n\).

When does such optimisation problem have feasible solutions (in the sense of a linear programming problem, i.e., solutions which satisfy the constraints)?

Intuitively, this should be possible if we are not demanding excessively large \(\gamma'_i\)s.

We now show that this is indeed the case when the spectral radius of matrix \(DF\) satisfies \(\rho(DF) < 1\).

Recall that the spectral radius \(\rho(X)\) of a matrix \(X\) is the largest absolute value of its eigenvalues.
Claim: If $A$ is a square matrix and if $\rho(A) < 1$ then $A^m \to 0$ (i.e., all entries of $A^m$ converge to 0)

This is easy to see if $A$ has $n$ linearly independent eigenvectors, because in this case $A$ can be represented as

$$A = Q\Lambda Q^{-1}$$

Here the $i^{th}$ column of $Q$ is the eigenvector corresponding to eigenvalue $\lambda_i$

$\Lambda$ is a diagonal matrix with $\lambda_i$ in the $i^{th}$ column and the $i^{th}$ row and zeros elsewhere.

In such a case

$$A^k = Q\Lambda Q^{-1}Q\Lambda Q^{-1} \ldots Q\Lambda Q^{-1} = Q\Lambda^k Q^{-1}.$$

Since $\Lambda^k$ has $\lambda_i^k$ on the diagonal, if $\rho(A) < 1$ then clearly $A^k \to 0$. 
Claim: If $A$ is a square matrix and if $\rho(A) < 1$ then $A^m \to 0$ (i.e., all entries of $A^m$ converge to 0)

This is easy to see if $A$ has $n$ linearly independent eigenvectors, because in this case $A$ can be represented as

$$A = Q\Lambda Q^{-1}$$

Here the $i^{th}$ column of $Q$ is the eigenvector corresponding to eigenvalue $\lambda_i$.

$\Lambda$ is a diagonal matrix with $\lambda_i$ in the $i^{th}$ column and the $i^{th}$ row and zeros elsewhere.

In such a case

$$A^k = Q\Lambda Q^{-1} Q\Lambda Q^{-1} \ldots Q\Lambda Q^{-1} Q\Lambda Q^{-1} = Q\Lambda^k Q^{-1}.$$ 

Since $\Lambda^k$ has $\lambda_i^k$ on the diagonal, if $\rho(A) < 1$ then clearly $A^k \to 0$. 
Claim: If $A$ is a square matrix and if $\rho(A) < 1$ then $A^m \to 0$ (i.e., all entries of $A^m$ converge to 0)

This is easy to see if $A$ has $n$ linearly independent eigenvectors, because in this case $A$ can be represented as

$$A = Q\Lambda Q^{-1}$$

Here the $i^{th}$ column of $Q$ is the eigenvector corresponding to eigenvalue $\lambda_i$

$\Lambda$ is a diagonal matrix with $\lambda_i$ in the $i^{th}$ column and the $i^{th}$ row and zeros elsewhere.

In such a case

$$A^k = Q\Lambda Q^{-1}Q\Lambda Q^{-1} \ldots Q\Lambda Q^{-1}Q\Lambda Q^{-1} = Q\Lambda^k Q^{-1}.$$  

Since $\Lambda^k$ has $\lambda_i^k$ on the diagonal, if $\rho(A) < 1$ then clearly $A^k \to 0$.  

Claim: If $A$ is a square matrix and if $\rho(A) < 1$ then $A^m \rightarrow 0$ (i.e., all entries of $A^m$ converge to 0)

This is easy to see if $A$ has $n$ linearly independent eigenvectors, because in this case $A$ can be represented as

$$A = Q\Lambda Q^{-1}$$

Here the $i^{th}$ column of $Q$ is the eigenvector corresponding to eigenvalue $\lambda_i$.

$\Lambda$ is a diagonal matrix with $\lambda_i$ in the $i^{th}$ column and the $i^{th}$ row and zeros elsewhere.

In such a case

$$A^k = Q\Lambda Q^{-1}Q\Lambda Q^{-1} \ldots Q\Lambda Q^{-1}Q\Lambda Q^{-1} = Q\Lambda^k Q^{-1}.$$  

Since $\Lambda^k$ has $\lambda_i^k$ on the diagonal, if $\rho(A) < 1$ then clearly $A^k \rightarrow 0$. 
Claim: If $A$ is a square matrix and if $\rho(A) < 1$ then $A^m \to 0$ (i.e., all entries of $A^m$ converge to 0)

This is easy to see if $A$ has $n$ linearly independent eigenvectors, because in this case $A$ can be represented as

$$A = Q\Lambda Q^{-1}$$

Here the $i^{th}$ column of $Q$ is the eigenvector corresponding to eigenvalue $\lambda_i$

$\Lambda$ is a diagonal matrix with $\lambda_i$ in the $i^{th}$ column and the $i^{th}$ row and zeros elsewhere.

In such a case

$$A^k = Q\Lambda Q^{-1} Q\Lambda Q^{-1} \ldots Q\Lambda Q^{-1} Q\Lambda Q^{-1} = Q\Lambda^k Q^{-1}.$$ 

Since $\Lambda^k$ has $\lambda_i^k$ on the diagonal, if $\rho(A) < 1$ then clearly $A^k \to 0.$
• **Claim:** If $A$ is a square matrix and if $\rho(A) < 1$ then $A^m \to 0$ (i.e., all entries of $A^m$ converge to 0)

• This is easy to see if $A$ has $n$ linearly independent eigenvectors, because in this case $A$ can be represented as

$$A = Q \Lambda Q^{-1}$$

• Here the $i^{th}$ column of $Q$ is the eigenvector corresponding to eigenvalue $\lambda_i$

• $\Lambda$ is a diagonal matrix with $\lambda_i$ in the $i^{th}$ column and the $i^{th}$ row and zeros elsewhere.

• In such a case

$$A^k = Q \Lambda Q^{-1} Q \Lambda Q^{-1} \ldots Q \Lambda Q^{-1} Q \Lambda Q^{-1} = Q \Lambda^k Q^{-1}.$$ 

• Since $\Lambda^k$ has $\lambda_i^k$ on the diagonal, if $\rho(A) < 1$ then clearly $A^k \to 0$. 
Moreover, it is easy to see that

\[(I - A) \sum_{i=0}^{k} A^i = \sum_{i=0}^{k} A^i - \sum_{i=1}^{k+1} A^i = I - A^{k+1}\]

Thus,

\[
\lim_{k \to \infty} \left( (I - A) \sum_{i=0}^{k} A^k \right) = \lim_{k \to \infty} (I - A^{k+1}),
\]

which implies

\[(I - A) \sum_{i=0}^{\infty} A^i = I.\]

This shows that matrix \( I - A \) is invertible and that

\[(I - A)^{-1} = \sum_{i=0}^{\infty} A^i\]

This is an important and extremely useful fact:

*If \( A \) is a square matrix and if \( \rho(A) < 1 \) then \( (I - A)^{-1} = \sum_{i=0}^{\infty} A^i \).*
Moreover, it is easy to see that

$$(I - A) \sum_{i=0}^{k} A^i = \sum_{i=0}^{k} A^i - \sum_{i=1}^{k+1} A^i = I - A^{k+1}$$

Thus,

$$\lim_{k \to \infty} \left( (I - A) \sum_{i=0}^{k} A^k \right) = \lim_{k \to \infty} (I - A^{k+1}),$$

which implies

$$(I - A) \sum_{i=0}^{\infty} A^i = I.$$

This shows that matrix $I - A$ is invertible and that

$$(I - A)^{-1} = \sum_{i=0}^{\infty} A^i$$

This is an important and extremely useful fact:

If $A$ is a square matrix and if $\rho(A) < 1$ then $(I - A)^{-1} = \sum_{i=0}^{\infty} A^i.$
Moreover, it is easy to see that
\[
(I - A) \sum_{i=0}^{k} A^i = \sum_{i=0}^{k} A^i - \sum_{i=1}^{k+1} A^i = I - A^{k+1}
\]

Thus,
\[
\lim_{k \to \infty} \left( (I - A) \sum_{i=0}^{k} A^k \right) = \lim_{k \to \infty} (I - A^{k+1})
\]
which implies
\[
(I - A) \sum_{i=0}^{\infty} A^i = I.
\]

This shows that matrix $I - A$ is invertible and that
\[
(I - A)^{-1} = \sum_{i=0}^{\infty} A^i
\]

This is an important and extremely useful fact:

*If $A$ is a square matrix and if $\rho(A) < 1$ then $(I - A)^{-1} = \sum_{i=0}^{\infty} A^i$.}*
Moreover, it is easy to see that

\[(I - A) \sum_{i=0}^{k} A^i = \sum_{i=0}^{k} A^i - \sum_{i=1}^{k+1} A^i = I - A^{k+1}\]

Thus,

\[\lim_{k \to \infty} \left( (I - A) \sum_{i=0}^{k} A^k \right) = \lim_{k \to \infty} (I - A^{k+1}),\]

which implies

\[(I - A) \sum_{i=0}^{\infty} A^i = I.\]

This shows that matrix \(I - A\) is invertible and that

\[(I - A)^{-1} = \sum_{i=0}^{\infty} A^i\]

This is an important and extremely useful fact:

*If \(A\) is a square matrix and if \(\rho(A) < 1\) then \((I - A)^{-1} = \sum_{i=0}^{\infty} A^i.\)*
Moreover, it is easy to see that

\[(I - A) \sum_{i=0}^{k} A^i = \sum_{i=0}^{k} A^i - \sum_{i=1}^{k+1} A^i = I - A^{k+1}\]

Thus,

\[
\lim_{k \to \infty} \left( (I - A) \sum_{i=0}^{k} A^k \right) = \lim_{k \to \infty} (I - A^{k+1}),
\]

which implies

\[(I - A) \sum_{i=0}^{\infty} A^i = I.\]

This shows that matrix \(I - A\) is invertible and that

\[(I - A)^{-1} = \sum_{i=0}^{\infty} A^i\]

This is an important and extremely useful fact:

*If \(A\) is a square matrix and if \(\rho(A) < 1\) then \((I - A)^{-1} = \sum_{i=0}^{\infty} A^i.*
Moreover, it is easy to see that

\[(I - A) \sum_{i=0}^{k} A^i = \sum_{i=0}^{k} A^i - \sum_{i=1}^{k+1} A^i = I - A^{k+1}\]

Thus,

\[\lim_{k \to \infty} \left( (I - A) \sum_{i=0}^{k} A^k \right) = \lim_{k \to \infty} (I - A^{k+1}),\]

which implies

\[(I - A) \sum_{i=0}^{\infty} A^i = I.\]

This shows that matrix \(I - A\) is invertible and that

\[(I - A)^{-1} = \sum_{i=0}^{\infty} A^i\]

This is an important and extremely useful fact:

*If \(A\) is a square matrix and if \(\rho(A) < 1\) then \((I - A)^{-1} = \sum_{i=0}^{\infty} A^i\).*
We now apply the above to matrix \( A = DF \).

Let \( p^* \) be given by

\[
p^* = (I - DF)^{-1}v = \sum_{i=0}^{\infty} (DF)^i v;
\]

Then \((I - DF)p^* = v\) and the constraint \((I - DF)p \geq v\) becomes

\[
(I - DF)p \geq (I - DF)p^*
\]

i.e.,

\[
(I - DF)(p - p^*) \geq 0
\]
We now apply the above to matrix $A = DF$.

Let $p^*$ be given by

$$p^* = (I - DF)^{-1}v = \sum_{i=0}^{\infty} (DF)^i v;$$

Then $(I - DF)p^* = v$ and the constraint

$$(I - DF)p \geq v$$

becomes

$$(I - DF)p \geq (I - DF)p^*$$

i.e.,

$$(I - DF)(p - p^*) \geq 0$$
- We now apply the above to matrix $A = DF$.

- Let $p^*$ be given by

$$p^* = (I - DF)^{-1}v = \sum_{i=0}^{\infty} (DF)^i v;$$

- Then $(I - DF)p^* = v$ and the constraint

$$(I - DF)p \geq v$$

becomes

$$(I - DF)(p - p^*) \geq 0$$

i.e.,
We now apply the above to matrix \( A = DF \).

Let \( \mathbf{p}^* \) be given by

\[
\mathbf{p}^* = (I - DF)^{-1}\mathbf{v} = \sum_{i=0}^{\infty} (DF)^i \mathbf{v};
\]

Then \( (I - DF)\mathbf{p}^* = \mathbf{v} \) and the constraint

\[
(I - DF)\mathbf{p} \geq \mathbf{v}
\]

becomes

\[
(I - DF')\mathbf{p} \geq (I - DF)\mathbf{p}^*
\]

i.e.,

\[
(I - DF)(\mathbf{p} - \mathbf{p}^*) \geq 0
\]
We now apply the above to matrix $A = DF$.

Let $\mathbf{p}^*$ be given by

$$
\mathbf{p}^* = (I - DF)^{-1}\mathbf{v} = \sum_{i=0}^{\infty} (DF)^i \mathbf{v};
$$

Then $(I - DF)\mathbf{p}^* = \mathbf{v}$ and the constraint

$$(I - DF)\mathbf{p} \geq \mathbf{v}$$

becomes

$$(I - DF')\mathbf{p} \geq (I - DF)\mathbf{p}^*$$

i.e.,

$$(I - DF)(\mathbf{p} - \mathbf{p}^*) \geq 0$$
Thus, all coordinates of vector \((I - DF)(p - p^*)\) are non-negative.

Since \((I - DF)^{-1} = \sum_{i=0}^{\infty} (DF)^k\) and all entries of matrix \(DF\) are non-negative, all entries of matrix \((I - DF)^{-1}\) are also all non-negative.

Thus, all coordinates of vector \((I - DF)^{-1}(I - DF)(p - p^*) = p - p^*\) are also non-negative.

However, \(p - p^* \geq 0\) (coordinate-wise) implies \(p \geq p^*\). Consequently, we conclude that \(1^T p\) attains its minimum for \(p = p^*\).
Thus, all coordinates of vector \((I - DF)(p - p^*)\) are non negative.

Since \((I - DF)^{-1} = \sum_{i=0}^{\infty} (DF)^k\) and all entries of matrix \(DF\) are non negative, all entries of matrix \((I - DF)^{-1}\) are also all non negative.

Thus, all coordinates of vector \((I - DF)^{-1}(I - DF)(p - p^*) = p - p^*\) are also non-negative.

However, \(p - p^* \geq 0\) (coordinate-wise) implies \(p \geq p^*\). Consequently, we conclude that \(1^T p\) attains its minimum for \(p = p^*\).
Thus, all coordinates of vector $\left( I - DF \right)(p - p^*)$ are non negative.

Since $\left( I - DF \right)^{-1} = \sum_{i=0}^{\infty} (DF)^k$ and all entries of matrix $DF$ are non negative, all entries of matrix $\left( I - DF \right)^{-1}$ are also all non negative.

Thus, all coordinates of vector $\left( I - DF \right)^{-1}(I - DF)(p - p^*) = p - p^*$ are also non-negative.

However, $p - p^* \geq 0$ (coordinate-wise) implies $p \geq p^*$. Consequently, we conclude that $1^Tp$ attains its minimum for $p = p^*$. 
Thus, all coordinates of vector \((I - DF)(p - p^*)\) are non negative.

Since \((I - DF)^{-1} = \sum_{i=0}^{\infty} (DF)^{k}\) and all entries of matrix \(DF\) are non negative, all entries of matrix \((I - DF)^{-1}\) are also all non negative.

Thus, all coordinates of vector \((I - DF)^{-1}(I - DF)(p - p^*) = p - p^*\) are also non-negative.

However, \(p - p^* \geq 0\) (coordinate-wise) implies \(p \geq p^*\). Consequently, we conclude that \(1^T p\) attains its minimum for \(p = p^*\).
This means that in order to minimise the sum total of all transmit power we have to find $\mathbf{p}^* = (I - DF)^{-1}\mathbf{v}$.

However, matrix inversion is numerically unstable, computationally expensive and cannot be done in a distributed manner without a communication overhead.

Luckily this is not necessary: since $\mathbf{p}^* = (I - DF)^{-1}\mathbf{v} = \sum_{i=0}^{\infty}(DF)^i\mathbf{v}$, we can approximate $\mathbf{p}^*$ in a similar way an approximation of the page rank was obtained:

Let $\mathbf{p}^{(0)}$ be an arbitrary vector, say giving all users the same initial power and define

$$\mathbf{p}^{(t)} = DF\mathbf{p}^{(t-1)} + \mathbf{v}.$$  

It is easy to show by induction that

$$\mathbf{p}^{(t)} = (DF)^t\mathbf{p}^{(0)} + \sum_{i=0}^{t-1}(DF)^i\mathbf{v}$$

This is because

$$\mathbf{p}^{(t+1)} = DF\mathbf{p}^{(t)} + \mathbf{v} = DF\left((DF)^t\mathbf{p}^{(0)} + \sum_{i=0}^{t-1}(DF)^i\mathbf{v}\right) + \mathbf{v}$$

$$= (DF)^{t+1}\mathbf{p}^{(0)} + \sum_{i=1}^{t}(DF)^i\mathbf{v} + \mathbf{v} = (DF)^{t+1}\mathbf{p}^{(0)} + \sum_{i=1}^{t}(DF)^i\mathbf{v}$$
This means that in order to minimise the sum total of all transmit power we have to find \( \mathbf{p}^* = (I - DF)^{-1}\mathbf{v} \).

However, matrix inversion is numerically unstable, computationally expensive and cannot be done in a distributed manner without a communication overhead.

Fortunately this is not necessary: since \( \mathbf{p}^* = (I - DF)^{-1}\mathbf{v} = \sum_{i=0}^{\infty} (DF)^i \mathbf{v} \), we can approximate \( \mathbf{p}^* \) in a similar way an approximation of the page rank was obtained.

Let \( \mathbf{p}^{(0)} \) be an arbitrary vector, say giving all users the same initial power and define

\[
\mathbf{p}^{(t)} = D F \mathbf{p}^{(t-1)} + \mathbf{v}.
\]

It is easy to show by induction that

\[
\mathbf{p}^{(t)} = (DF)^t \mathbf{p}^{(0)} + \sum_{i=0}^{t-1} (DF)^i \mathbf{v}
\]

This is because

\[
\mathbf{p}^{(t+1)} = D F \mathbf{p}^{(t)} + \mathbf{v} = D F \left( (DF)^t \mathbf{p}^{(0)} + \sum_{i=0}^{t-1} (DF)^i \mathbf{v} \right) + \mathbf{v}
\]

\[
= (DF)^{t+1} \mathbf{p}^{(0)} + \sum_{i=1}^{t} (DF)^i \mathbf{v} + \mathbf{v} = (DF)^{t+1} \mathbf{p}^{(0)} + \sum_{i=1}^{t} (DF)^i \mathbf{v}
\]
This means that in order to minimise the sum total of all transmit power we have to find \( p^* = (I - DF)^{-1}v \).

However, matrix inversion is numerically unstable, computationally expensive and cannot be done in a distributed manner without a communication overhead.

Luckily this is not necessary: since \( p^* = (I - DF)^{-1}v = \sum_{i=0}^{\infty} (DF)^i v \), we can approximate \( p^* \) in a similar way an approximation of the page rank was obtained:

Let \( p(0) \) be an arbitrary vector, say giving all users the same initial power and define

\[
p(t) = DFp(t-1) + v.
\]

It is easy to show by induction that

\[
p(t) = (DF)^t p(0) + \sum_{i=0}^{t-1} (DF)^i v
\]

This is because

\[
p(t+1) = DFp(t) + v = DF \left( (DF)^t p(0) + \sum_{i=0}^{t-1} (DF)^i v \right) + v
\]

\[
= (DF)^{t+1} p(0) + \sum_{i=1}^{t} (DF)^i v + v = (DF)^{t+1} p(0) + \sum_{i=1}^{t} (DF)^i v
\]
This means that in order to minimise the sum total of all transmit power we have to find \( p^* = (I - DF)^{-1}v \).

However, matrix inversion is numerically unstable, computationally expensive and cannot be done in a distributed manner without a communication overhead.

Luckily this is not necessary: since \( p^* = (I - DF)^{-1}v = \sum_{i=0}^{\infty} (DF)^i v \), we can approximate \( p^* \) in a similar way an approximation of the page rank was obtained:

Let \( p(0) \) be an arbitrary vector, say giving all users the same initial power and define

\[
p(t) = DFp(t-1) + v.
\]

It is easy to show by induction that

\[
p(t) = (DF)^t p(0) + \sum_{i=0}^{t-1} (DF)^i v
\]

This is because

\[
p(t+1) = DFp(t) + v = DF \left( (DF)^t p(0) + \sum_{i=0}^{t-1} (DF)^i v \right) + v
\]

\[
= (DF)^{t+1} p(0) + \sum_{i=1}^{t} (DF)^i v + v = (DF)^{t+1} p(0) + \sum_{i=1}^{t} (DF)^i v
\]
This means that in order to minimise the sum total of all transmit power we have to find \( p^* = (I - DF)^{-1}v \).

However, matrix inversion is numerically unstable, computationally expensive and cannot be done in a distributed manner without a communication overhead.

Luckily this is not necessary: since \( p^* = (I - DF)^{-1}v = \sum_{i=0}^{\infty} (DF)^i v \), we can approximate \( p^* \) in a similar way an approximation of the page rank was obtained:

Let \( p^{(0)} \) be an arbitrary vector, say giving all users the same initial power and define

\[
p^{(t)} = DF p^{(t-1)} + v.
\]

It is easy to show by induction that

\[
p^{(t)} = (DF)^t p^{(0)} + \sum_{i=0}^{t-1} (DF)^i v
\]

This is because

\[
p^{(t+1)} = DF p^{(t)} + v = DF \left( (DF)^t p^{(0)} + \sum_{i=0}^{t-1} (DF)^i v \right) + v = (DF)^{t+1} p^{(0)} + \sum_{i=1}^{t} (DF)^i v + v = (DF)^{t+1} p^{(0)} + \sum_{i=1}^{t} (DF)^i v
\]
This means that in order to minimise the sum total of all transmit power we have to find $p^* = (I - DF)^{-1}v$.

However, matrix inversion is numerically unstable, computationally expensive and cannot be done in a distributed manner without a communication overhead.

Luckily this is not necessary: since $p^* = (I - DF)^{-1}v = \sum_{i=0}^{\infty} (DF)^i v$, we can approximate $p^*$ in a similar way an approximation of the page rank was obtained:

Let $p^{(0)}$ be an arbitrary vector, say giving all users the same initial power and define

$$p^{(t)} = DFp^{(t-1)} + v.$$ 

It is easy to show by induction that

$$p^{(t)} = (DF)^tp^{(0)} + \sum_{i=0}^{t-1} (DF)^i v$$

This is because

$$p^{(t+1)} = DFp^{(t)} + v = DF \left( (DF)^tp^{(0)} + \sum_{i=0}^{t-1} (DF)^i v \right) + v$$

$$= (DF)^{t+1}p^{(0)} + \sum_{i=1}^{t} (DF)^i v + v = (DF)^{t+1}p^{(0)} + \sum_{i=0}^{t} (DF)^i v$$
Since \((DF)^t p^{(0)} \to 0\) we obtain from

\[
p^{(t)} = (DF)^t p^{(0)} + \sum_{i=0}^{t-1} (DF)^i v
\]

that \(p^{(t)} \to \sum_{i=0}^{\infty} (DF)^i v = p^*\).

However, the recursive update

\[
p^{(t+1)} = DF p^{(t)} + v
\]

when expressed through its coordinates becomes

\[
p_{i}^{(t+1)} = \gamma_i \sum_{j:j \neq i} \frac{G_{ij}}{G_{ii}} p_j^{(t)} + \gamma_i \eta_i = \sum_{j:j \neq i} \frac{G_{ij} p_j^{(t)} + \eta_i}{G_{ii}}
\]

\[
= \frac{\gamma_i p_i^{(t)}}{p_i^{(t)} G_{ii}} \sum_{j:j \neq i} \frac{G_{ij} p_j^{(t)}}{p_j^{(t)} G_{ii}} + \eta_i = \frac{\gamma_i}{SIR_i^{(t)}} p_i^{(t)}
\]

But this is exactly our recursive update formula.
Since \((DF)^t p^{(0)} \to 0\) we obtain from

\[
p^{(t)} = (DF)^t p^{(0)} + \sum_{i=0}^{t-1} (DF)^i v
\]

that \(p^{(t)} \to \sum_{i=0}^{\infty} (DF)^i v = p^*\).

However, the recursive update

\[
p^{(t+1)} = DFp^{(t)} + v
\]

when expressed through its coordinates becomes

\[
p_i^{(t+1)} = \gamma_i \sum_{j:j \neq i} \frac{G_{ij} p_j^{(t)}}{G_{ii}} + \frac{\gamma_i \eta_i}{G_{ii}} = \gamma_i \sum_{j:j \neq i} \frac{G_{ij} p_j^{(t)}}{G_{ii}} + \eta_i
\]

\[
= \gamma_i p_i^{(t)} \frac{\sum_{j:j \neq i} G_{ij} p_j^{(t)} + \eta_i}{p_i^{(t)} G_{ii}} = \frac{\gamma_i}{SIR_i^{(t)}} p_i^{(t)}
\]

But this is exactly our recursive update formula.
Since \((DF)^t p^{(0)} \rightarrow 0\) we obtain from

\[ p^{(t)} = (DF)^t p^{(0)} + \sum_{i=0}^{t-1} (DF)^i v \]

that \(p^{(t)} \rightarrow \sum_{i=0}^{\infty} (DF)^i v = p^*\).

However, the recursive update

\[ p^{(t+1)} = DF p^{(t)} + v \]

when expressed through its coordinates becomes

\[ p^{(t+1)}_i = \gamma_i \sum_{j:j \neq i} \frac{G_{ij} p^{(t)}_j}{G_{ii}} + \frac{\gamma_i \eta_i}{G_{ii}} = \gamma_i \sum_{j:j \neq i} \frac{G_{ij} p^{(t)}_j}{G_{ii}} + \eta_i \]

\[ = \gamma_i p^{(t)}_i \sum_{j:j \neq i} \frac{G_{ij} p^{(t)}_j}{p^{(t)}_i G_{ii}} + \eta_i = \frac{\gamma_i}{SIR^{(t)}_i} p^{(t)}_i \]

But this is exactly our recursive update formula.
Since \((DF)^t p^{(0)} \to 0\) we obtain from

\[
p^{(t)} = (DF)^t p^{(0)} + \sum_{i=0}^{t-1} (DF)^i v
\]

that \(p^{(t)} \to \sum_{i=0}^{\infty} (DF)^i v = p^*\).

However, the recursive update

\[
p^{(t+1)} = DF p^{(t)} + v
\]

when expressed through its coordinates becomes

\[
p_{i}^{(t+1)} = \gamma_i \sum_{j: j \neq i} \frac{G_{ij}}{G_{ii}} p_j^{(t)} + \gamma_i \eta_i = \gamma_i \frac{\sum_{j: j \neq i} G_{ij} p_j^{(t)} + \eta_i}{G_{ii}}
\]

\[
= \gamma_i p_i^{(t)} \frac{\sum_{j: j \neq i} G_{ij} p_j^{(t)} + \eta_i}{p_i^{(t)} G_{ii}} = \frac{\gamma_i}{SIR_i^{(t)}} p_i^{(t)}
\]

But this is exactly our recursive update formula.
Thus, such update make the powers continuously chase the optimal solution, even as the situation changes by adding new participants and dropping some of the old ones and with the change in the noise of the environment.

But how should we chose the target values $\gamma_i$ which might be different for different users having different equipment and different bandwidth needs?

In order to optimise the bandwidth use, the communication takes place close to the information theoretic capacity of the channel, which is determined by the signal to noise ratio. Essentially, the distances between any two symbols (two short waveforms) must be greater than the RMS value of the noise so that the received signal has a chance to be correctly recognised most of the time.

However, such distances cannot be significantly larger than that, because this would limit the number of distinct signals within certain power envelope, and would thus significantly reduce the available capacity. Thus, even if most of the time the noise will not cause a symbol to be misinterpreted as a wrong, neighbouring symbol, there always will be a certain probability that in fact some of the received symbols will be incorrectly identified.

Rather than having to repeat messages, transceivers rely on certain small level of redundancy which allows the entire message to be perfectly reconstructed despite the fact that a few symbols in the message might have been misread.
Thus, such update make the powers continuously chase the optimal solution, even as the situation changes by adding new participants and dropping some of the old ones and with the change in the noise of the environment.

But how should we chose the target values $\gamma_i$ which might be different for different users having different equipment and different bandwidth needs?

In order to optimise the bandwidth use, the communication takes place close to the information theoretic capacity of the channel, which is determined by the signal to noise ratio. Essentially, the distances between any two symbols (two short waveforms) must be greater than the RMS value of the noise so that the received signal has a chance to be correctly recognised most of the time.

However, such distances cannot be significantly larger than that, because this would limit the number of distinct signals within certain power envelope, and would thus significantly reduce the available capacity. Thus, even if most of the time the noise will not cause a symbol to be misinterpreted as a wrong, neighbouring symbol, there always will be a certain probability that in fact some of the received symbols will be incorrectly identified.

Rather than having to repeat messages, transceivers rely on certain small level of redundancy which allows the entire message to be perfectly reconstructed despite the fact that a few symbols in the message might have been misread.
Thus, such update make the powers continuously chase the optimal solution, even as the situation changes by adding new participants and dropping some of the old ones and with the change in the noise of the environment.

But how should we chose the target values $\gamma_i$ which might be different for different users having different equipment and different bandwidth needs?

In order to optimise the bandwidth use, the communication takes place close to the information theoretic capacity of the channel, which is determined by the signal to noise ratio. Essentially, the distances between any two symbols (two short waveforms) must be greater than the RMS value of the noise so that the received signal has a chance to be correctly recognised most of the time.

However, such distances cannot be significantly larger than that, because this would limit the number of distinct signals within certain power envelope, and would thus significantly reduce the available capacity. Thus, even if most of the time the noise will not cause a symbol to be misinterpreted as a wrong, neighbouring symbol, there always will be a certain probability that in fact some of the received symbols will be incorrectly identified.

Rather than having to repeat messages, transceivers rely on certain small level of redundancy which allows the entire message to be perfectly reconstructed despite the fact that a few symbols in the message might have been misread.
Thus, such update make the powers continuously chase the optimal solution, even as the situation changes by adding new participants and dropping some of the old ones and with the change in the noise of the environment.

But how should we chose the target values $\gamma_i$ which might be different for different users having different equipment and different bandwidth needs?

In order to optimise the bandwidth use, the communication takes place close to the information theoretic capacity of the channel, which is determined by the signal to noise ratio. Essentially, the distances between any two symbols (two short waveforms) must be greater than the RMS value of the noise so that the received signal has a chance to be correctly recognised most of the time.

However, such distances cannot be significantly larger than that, because this would limit the number of distinct signals within certain power envelope, and would thus significantly reduce the available capacity. Thus, even if most of the time the noise will not cause a symbol to be misinterpreted as a wrong, neighbouring symbol, there always will be a certain probability that in fact some of the received symbols will be incorrectly identified.

Rather than having to repeat messages, transceivers rely on certain small level of redundancy which allows the entire message to be perfectly reconstructed despite the fact that a few symbols in the message might have been misread.
Thus, such update make the powers continuously chase the optimal solution, even as the situation changes by adding new participants and dropping some of the old ones and with the change in the noise of the environment.

But how should we chose the target values $\gamma_i$ which might be different for different users having different equipment and different bandwidth needs?

In order to optimise the bandwidth use, the communication takes place close to the information theoretic capacity of the channel, which is determined by the signal to noise ratio. Essentially, the distances between any two symbols (two short waveforms) must be greater than the RMS value of the noise so that the received signal has a chance to be correctly recognised most of the time.

However, such distances cannot be significantly larger than that, because this would limit the number of distinct signals within certain power envelope, and would thus significantly reduce the available capacity. Thus, even if most of the time the noise will not cause a symbol to be misinterpreted as a wrong, neighbouring symbol, there always will be a certain probability that in fact some of the received symbols will be incorrectly identified.

Rather than having to repeat messages, transceivers rely on certain small level of redundancy which allows the entire message to be perfectly reconstructed despite the fact that a few symbols in the message might have been misread.
In other words, the encoding is done in such a way that it incorporates certain level of error correction. If one wishes to send say a message consisting of 200 symbols, one might actually send 256 symbols but chosen in such a way that up to 28 symbols can be misinterpreted due to noise, and yet the entire original message of 200 symbols can be perfectly reconstructed.

We will see in a moment how this is done; here we first explain how error correction codes are used for the outer loop of transmit power control, i.e., for choosing the target $\gamma_i$ for a user $i$.

The idea is again very simple: if the error correction can fix up to, say, 10% of symbols wrongly received, then $\gamma_i$ will be chosen so that the number of errors is somewhat below 10%, say 5%.

If the errors are lower than 5%, the corresponding $\gamma_i$ will be proportionally decreased; if the errors go above 5%, the corresponding target $\gamma_i$ will be proportionally increased.
In other words, the encoding is done in such a way that it incorporates certain level of error correction. If one wishes to send say a message consisting of 200 symbols, one might actually send 256 symbols but chosen in such a way that up to 28 symbols can be misinterpreted due to noise, and yet the entire original message of 200 symbols can be perfectly reconstructed.

We will see in a moment how this is done; here we first explain how error correction codes are used for the outer loop of transmit power control, i.e., for choosing the target $\gamma_i$ for a user $i$.

The idea is again very simple: if the error correction can fix up to, say, 10% of symbols wrongly received, then $\gamma_i$ will be chosen so that the number of errors is somewhat below 10%, say 5%.

If the errors are lower than 5%, the corresponding $\gamma_i$ will be proportionally decreased; if the errors go above 5%, the corresponding target $\gamma_i$ will be proportionally increased.
In other words, the encoding is done in such a way that it incorporates certain level of error correction. If one wishes to send say a message consisting of 200 symbols, one might actually send 256 symbols but chosen in such a way that up to 28 symbols can be misinterpreted due to noise, and yet the entire original message of 200 symbols can be perfectly reconstructed.

We will see in a moment how this is done; here we first explain how error correction codes are used for the outer loop of transmit power control, i.e., for choosing the target $\gamma_i$ for a user $i$.

The idea is again very simple: if the error correction can fix up to, say, 10% of symbols wrongly received, then $\gamma_i$ will be chosen so that the number of errors is somewhat below 10%, say 5%.

If the errors are lower than 5%, the corresponding $\gamma_i$ will be proportionally decreased; if the errors go above 5%, the corresponding target $\gamma_i$ will be proportionally increased.
In other words, the encoding is done in such a way that it incorporates certain level of error correction. If one wishes to send say a message consisting of 200 symbols, one might actually send 256 symbols but chosen in such a way that up to 28 symbols can be misinterpreted due to noise, and yet the entire original message of 200 symbols can be perfectly reconstructed.

We will see in a moment how this is done; here we first explain how error correction codes are used for the outer loop of transmit power control, i.e., for choosing the target $\gamma_i$ for a user $i$.

The idea is again very simple: if the error correction can fix up to, say, 10% of symbols wrongly received, then $\gamma_i$ will be chosen so that the number of errors is somewhat below 10%, say 5%.

If the errors are lower than 5%, the corresponding $\gamma_i$ will be proportionally decreased; if the errors go above 5%, the corresponding target $\gamma_i$ will be proportionally increased.
We now recall the Lagrange interpolation formula, which allows us to reconstruct a polynomial of degree \( n - 1 \) if we are given its values \( v_1, v_2, \ldots, v_n \) for \( n \) distinct inputs \( c_1, c_2, \ldots, c_n \):

\[
P(x) = \sum_{i=1}^{n} \frac{\prod_{j:j \neq i} (x - c_j)}{\prod_{j:j \neq i} (c_i - c_j)} v_i
\]

Note that the polynomials

\[
P_i(x) = \frac{\prod_{j:j \neq i} (x - c_j)}{\prod_{j:j \neq i} (c_i - c_j)}
\]

have the property that \( P_i(c_k) = 0 \) if \( k \neq i \) and \( P_i(c_i) = 1 \).
We now recall the Lagrange interpolation formula, which allows us to reconstruct a polynomial of degree $n - 1$ if we are given its values $v_1, v_2, \ldots, v_n$ for $n$ distinct inputs $c_1, c_2, \ldots, c_n$:

$$P(x) = \sum_{i=1}^{n} \frac{\prod_{j:j \neq i}(x - c_j)}{\prod_{j:j \neq i}(c_i - c_j)} v_i$$

Note that the polynomials

$$P_i(x) = \frac{\prod_{j:j \neq i}(x - c_j)}{\prod_{j:j \neq i}(c_i - c_j)}$$

have the property that $P_i(c_k) = 0$ if $k \neq i$ and $P_i(c_i) = 1$. 
Assume we want to send a message \( m \) which is a sequence of 10 symbols which are just 10 integers, but we suspect that, due to noise in the communication channel, one of the sent symbols might be wrongly received as another integer.

An obvious way to make it far more likely that all the symbols (integers) will be received correctly would be to repeat each symbol three times and take the majority vote at the receiving end.

However, this produces a very large overhead, causing us to send 30 numbers in order to increase the likelihood that the original 10 numbers will be correctly received.

We now show how, in principle, we can make our transmission more noise robust, by sending only 12 numbers, so that even if one of them gets corrupted, we can still recover the sent message perfectly.
Assume we want to send a message $m$ which is a sequence of 10 symbols which are just 10 integers, but we suspect that, due to noise in the communication channel, one of the sent symbols might be wrongly received as another integer.

An obvious way to make it far more likely that all the symbols (integers) will be received correctly would be to repeat each symbol three times and take the majority vote at the receiving end.

However, this produces a very large overhead, causing us to send 30 numbers in order to increase the likelihood that the original 10 numbers will be correctly received.

We now show how, in principle, we can make our transmission more noise robust, by sending only 12 numbers, so that even if one of them gets corrupted, we can still recover the sent message perfectly.
Assume we want to send a message $m$ which is a sequence of 10 symbols which are just 10 integers, but we suspect that, due to noise in the communication channel, one of the sent symbols might be wrongly received as another integer.

An obvious way to make it far more likely that all the symbols (integers) will be received correctly would be to repeat each symbol three times and take the majority vote at the receiving end.

However, this produces a very large overhead, causing us to send 30 numbers in order to increase the likelihood that the original 10 numbers will be correctly received.

We now show how, in principle, we can make our transmission more noise robust, by sending only 12 numbers, so that even if one of them gets corrupted, we can still recover the sent message perfectly.
Assume we want to send a message $m$ which is a sequence of 10 symbols which are just 10 integers, but we suspect that, due to noise in the communication channel, one of the sent symbols might be wrongly received as another integer.

An obvious way to make it far more likely that all the symbols (integers) will be received correctly would be to repeat each symbol three times and take the majority vote at the receiving end.

However, this produces a very large overhead, causing us to send 30 numbers in order to increase the likelihood that the original 10 numbers will be correctly received.

We now show how, in principle, we can make our transmission more noise robust, by sending only 12 numbers, so that even if one of them gets corrupted, we can still recover the sent message perfectly.
To make the communication more robust, if we want to transmit a sequence of ten numbers $\mathbf{a}^T = \langle a_1, a_2, \ldots, a_{10} \rangle$ we first form the following polynomial:

$$P(x) = a_{10}x^9 + a_9x^8 + \ldots + a_2x + a_1,$$  \hspace{1cm} (4)

Instead of transmitting the original sequence $\mathbf{a}^T = \langle a_1, a_2, \ldots, a_{10} \rangle$ we transmit its code which is a sequence of 12 values of this polynomial, i.e.,

$\mathbf{b}^T = \langle b_1, b_2, \ldots, b_{10}, b_{11}, b_{12} \rangle = \langle P(1), P(2), \ldots, P(10), P(11), P(12) \rangle$.

Assume now that one of these values $b_m$ which is supposed to be $P(m)$ gets corrupted and is received as another, wrong value instead, but of course we do not know which one is the wrong one, i.e., we do not know what $m$ is.

We can now still perfectly decode the message using the fact that among these 12 numbers we can find a unique subsequence of 11 numbers that are the values of the same polynomial of degree 9.

Thus, to decode our message, we try 12 subsequences each containing 11 out of 12 received numbers, and for each such subsequence of 11 numbers we pick arbitrary 10 of them, construct the corresponding interpolation polynomial and check if the 11\textsuperscript{th} number is also the correct value the same polynomial.
To make the communication more robust, if we want to transmit a sequence of ten numbers $\mathbf{a}^T = \langle a_1, a_2, \ldots, a_{10} \rangle$ we first form the following polynomial:

$$P(x) = a_{10}x^9 + a_9x^8 + \ldots + a_2x + a_1,$$

(4)

Instead of transmitting the original sequence $\mathbf{a}^T = \langle a_1, a_2, \ldots, a_{10} \rangle$ we transmit its code which is a sequence of 12 values of this polynomial, i.e., $\mathbf{b}^T = \langle b_1, b_2, \ldots, b_{10}, b_{11}, b_{12} \rangle = \langle P(1), P(2), \ldots, P(10), P(11), P(12) \rangle$.

Assume now that one of these values $b_m$ which is supposed to be $P(m)$ gets corrupted and is received as another, wrong value instead, but of course we do not know which one is the wrong one, i.e., we do not know what $m$ is.

We can now still perfectly decode the message using the fact that among these 12 numbers we can find a unique subsequence of 11 numbers that are the values of the same polynomial of degree 9.

Thus, to decode our message, we try 12 subsequences each containing 11 out of 12 received numbers, and for each such subsequence of 11 numbers we pick arbitrary 10 of them, construct the corresponding interpolation polynomial and check if the 11th number is also the correct value the same polynomial.
To make the communication more robust, if we want to transmit a sequence of ten numbers $\mathbf{a}^T = \langle a_1, a_2, \ldots, a_{10} \rangle$ we first form the following polynomial:

$$ P(x) = a_{10}x^{9} + a_{9}x^{8} + \ldots + a_{2}x + a_{1}, $$

(4)

Instead of transmitting the original sequence $\mathbf{a}^T = \langle a_1, a_2, \ldots, a_{10} \rangle$ we transmit its code which is a sequence of 12 values of this polynomial, i.e.,

$\mathbf{b}^T = \langle b_1, b_2, \ldots, b_{10}, b_{11}, b_{12} \rangle = \langle P(1), P(2), \ldots, P(10), P(11), P(12) \rangle$.

Assume now that one of these values $b_m$ which is supposed to be $P(m)$ gets corrupted and is received as another, wrong value instead, but of course we do not know which one is the wrong one, i.e., we do not know what $m$ is.

We can now still perfectly decode the message using the fact that among these 12 numbers we can find a unique subsequence of 11 numbers that are the values of the same polynomial of degree 9.

Thus, to decode our message, we try 12 subsequences each containing 11 out of 12 received numbers, and for each such subsequence of 11 numbers we pick arbitrary 10 of them, construct the corresponding interpolation polynomial and check if the 11th number is also the correct value the same polynomial.
To make the communication more robust, if we want to transmit a sequence of ten numbers \( \mathbf{a}^T = \langle a_1, a_2, \ldots, a_{10} \rangle \) we first form the following polynomial:

\[
P(x) = a_{10} x^9 + a_9 x^8 + \ldots + a_2 x + a_1,
\]

(4)

Instead of transmitting the original sequence \( \mathbf{a}^T = \langle a_1, a_2, \ldots, a_{10} \rangle \) we transmit its code which is a sequence of 12 values of this polynomial, i.e., \( \mathbf{b}^T = \langle b_1, b_2, \ldots, b_{10}, b_{11}, b_{12} \rangle = \langle P(1), P(2), \ldots, P(10), P(11), P(12) \rangle \).

Assume now that one of these values \( b_m \) which is supposed to be \( P(m) \) gets corrupted and is received as another, wrong value instead, but of course we do not know which one is the wrong one, i.e., we do not know what \( m \) is.

We can now still perfectly decode the message using the fact that among these 12 numbers we can find a unique subsequence of 11 numbers that are the values of the same polynomial of degree 9.

Thus, to decode our message, we try 12 subsequences each containing 11 out of 12 received numbers, and for each such subsequence of 11 numbers we pick arbitrary 10 of them, construct the corresponding interpolation polynomial and check if the 11\(^{th}\) number is also the correct value the same polynomial.
To make the communication more robust, if we want to transmit a sequence of ten numbers $\mathbf{a}^T = \langle a_1, a_2, \ldots, a_{10} \rangle$ we first form the following polynomial:

$$P(x) = a_{10}x^9 + a_9x^8 + \ldots + a_2x + a_1,$$

Instead of transmitting the original sequence $\mathbf{a}^T = \langle a_1, a_2, \ldots, a_{10} \rangle$ we transmit its code which is a sequence of 12 values of this polynomial, i.e., $\mathbf{b}^T = \langle b_1, b_2, \ldots, b_{10}, b_{11}, b_{12} \rangle = \langle P(1), P(2), \ldots, P(10), P(11), P(12) \rangle$.

Assume now that one of these values $b_m$ which is supposed to be $P(m)$ gets corrupted and is received as another, wrong value instead, but of course we do not know which one is the wrong one, i.e., we do not know what $m$ is.

We can now still perfectly decode the message using the fact that among these 12 numbers we can find a unique subsequence of 11 numbers that are the values of the same polynomial of degree 9.

Thus, to decode our message, we try 12 subsequences each containing 11 out of 12 received numbers, and for each such subsequence of 11 numbers we pick arbitrary 10 of them, construct the corresponding interpolation polynomial and check if the 11th number is also the correct value the same polynomial.
For example, we can start with the subsequence \(b_1, b_2, \ldots b_{11}\), form the polynomial

\[
Q(x) = \sum_{i=1}^{10} \frac{\prod_{j \neq i} (x - j)}{\prod_{j \neq i} (i - j)} b_i
\]

and then check if \(Q(11) = b_{11}\).

Note that for any set of 11 values, since at most one is wrong, at least ten will be right, and there is only one polynomial of degree 9 with values equal to these 10 values, so only \(P(x)\) will fit these 10 values and no other polynomial.

Clearly, checking all subsequences with 11 elements we will eventually find one that is a sequence of 11 values of the same polynomial \(Q(x)\) and the coefficients of that polynomial reproduce the correct message, because, as we have just explained, only the original polynomial \(P(x)\) can fit 10 correct values of \(P(x)\).
For example, we can start with the subsequence $b_1, b_2, \ldots b_{11}$, form the polynomial

\[
Q(x) = \sum_{i=1}^{10} \frac{\prod_{j \neq i}(x - j)}{\prod_{j \neq i}(i - j)} b_i
\]

and then check if $Q(11) = b_{11}$.

Note that for any set of 11 values, since at most one is wrong, at least ten will be right, and there is only one polynomial of degree 9 with values equal to these 10 values, so only $P(x)$ will fit these 10 values and no other polynomial.

Clearly, checking all subsequences with 11 elements we will eventually find one that is a sequence of 11 values of the same polynomial $Q(x)$ and the coefficients of that polynomial reproduce the correct message, because, as we have just explained, only the original polynomial $P(x)$ can fit 10 correct values of $P(x)$. 
For example, we can start with the subsequence $b_1, b_2, \ldots b_{11}$, form the polynomial

$$Q(x) = \sum_{i=1}^{10} \frac{\prod_{j \neq i} (x - j)}{\prod_{j \neq i} (i - j)} b_i$$

and then check if $Q(11) = b_{11}$.

Note that for any set of 11 values, since at most one is wrong, at least ten will be right, and there is only one polynomial of degree 9 with values equal to these 10 values, so only $P(x)$ will fit these 10 values and no other polynomial.

Clearly, checking all subsequences with 11 elements we will eventually find one that is a sequence of 11 values of the same polynomial $Q(x)$ and the coefficients of that polynomial reproduce the correct message, because, as we have just explained, only the original polynomial $P(x)$ can fit 10 correct values of $P(x)$.
Now lets assume that the channel can produce up to two errors. How many values of the same polynomial $P(x)$ of degree 9, given by (4) do we now need? Would 13 values be enough?

We could try searching through all subsequences of $P(1), \ldots, P(13)$ of size 11, picking out of them arbitrary 10 values and checking if the $11^{th}$ value satisfies the same polynomial of degree 9.

Unfortunately this does not work: even though unlikely, the following situation might happen.

Say we picked $b_1, b_2, \ldots, b_{10}, b_{11}$ and form an interpolation polynomial $Q(x)$ for values $b_1, b_2, \ldots, b_{10}$.

Assume also that it just happens that the two corrupt values are $b_{10}$ and $b_{11}$.

Thus, $Q(x)$ involves one corrupt value $b_{10}$ and so $Q(x) \neq P(x)$. However, by bad luck, it just happens that $b_{11}$ is corrupted in such a way that $Q(11) = b_{11}$ and we wrongly conclude that the right polynomial is $Q(x)$. 
Now let's assume that the channel can produce up to two errors. How many values of the same polynomial \( P(x) \) of degree 9, given by (4) do we now need? Would 13 values be enough?

We could try searching through all subsequences of \( P(1), \ldots, P(13) \) of size 11, picking out of them arbitrary 10 values and checking if the 11\(^{th}\) value satisfies the same polynomial of degree 9.

Unfortunately this does not work: even though unlikely, the following situation might happen.

Say we picked \( b_1, b_2, \ldots, b_{10}, b_{11} \) and form an interpolation polynomial \( Q(x) \) for values \( b_1, b_2, \ldots, b_{10} \).

Assume also that it just happens that the two corrupt values are \( b_{10} \) and \( b_{11} \).

Thus, \( Q(x) \) involves one corrupt value \( b_{10} \) and so \( Q(x) \neq P(x) \). However, by bad luck, it just happens that \( b_{11} \) is corrupted in such a way that \( Q(11) = b_{11} \) and we wrongly conclude that the right polynomial is \( Q(x) \).
Now let's assume that the channel can produce up to two errors. How many values of the same polynomial $P(x)$ of degree 9, given by (4) do we now need? Would 13 values be enough?

We could try searching through all subsequences of $P(1), \ldots, P(13)$ of size 11, picking out of them arbitrary 10 values and checking if the 11th value satisfies the same polynomial of degree 9.

Unfortunately this does not work: even though unlikely, the following situation might happen.

Say we picked $b_1, b_2, \ldots, b_{10}, b_{11}$ and form an interpolation polynomial $Q(x)$ for values $b_1, b_2, \ldots, b_{10}$.

Assume also that it just happens that the two corrupt values are $b_{10}$ and $b_{11}$.

Thus, $Q(x)$ involves one corrupt value $b_{10}$ and so $Q(x) \neq P(x)$. However, by bad luck, it just happens that $b_{11}$ is corrupted in such a way that $Q(11) = b_{11}$ and we wrongly conclude that the right polynomial is $Q(x)$. 
Now lets assume that the channel can produce up to two errors. How many values of the same polynomial \( P(x) \) of degree 9, given by (4) do we now need? Would 13 values be enough?

We could try searching through all subsequences of \( P(1), \ldots, P(13) \) of size 11, picking out of them arbitrary 10 values and checking if the 11\(^{th}\) value satisfies the same polynomial of degree 9.

Unfortunately this does not work: even though unlikely, the following situation might happen.

Say we picked \( b_1, b_2, \ldots, b_{10}, b_{11} \) and form an interpolation polynomial \( Q(x) \) for values \( b_1, b_2, \ldots, b_{10} \).

Assume also that it just happens that the two corrupt values are \( b_{10} \) and \( b_{11} \).

Thus, \( Q(x) \) involves one corrupt value \( b_{10} \) and so \( Q(x) \neq P(x) \). However, by bad luck, it just happens that \( b_{11} \) is corrupted in such a way that \( Q(11) = b_{11} \) and we wrongly conclude that the right polynomial is \( Q(x) \).
Now let's assume that the channel can produce up to two errors. How many values of the same polynomial \( P(x) \) of degree 9, given by (4) do we now need? Would 13 values be enough?

We could try searching through all subsequences of \( P(1), \ldots, P(13) \) of size 11, picking out of them arbitrary 10 values and checking if the 11\(^{th}\) value satisfies the same polynomial of degree 9.

Unfortunately this does not work: even though unlikely, the following situation might happen.

Say we picked \( b_1, b_2, \ldots, b_{10}, b_{11} \) and form an interpolation polynomial \( Q(x) \) for values \( b_1, b_2, \ldots, b_{10} \).

Assume also that it just happens that the two corrupt values are \( b_{10} \) and \( b_{11} \).

Thus, \( Q(x) \) involves one corrupt value \( b_{10} \) and so \( Q(x) \neq P(x) \). However, by bad luck, it just happens that \( b_{11} \) is corrupted in such a way that \( Q(11) = b_{11} \) and we wrongly conclude that the right polynomial is \( Q(x) \).
Let us now try sending 14 values instead.

Since at most 2 values are corrupted, we do know that there must be at least 12 values which all correspond to a polynomial of degree 9 - at least one such polynomial is just $P(x)$.

However, if any 12 values correspond to a same polynomial, say $Q(x)$, since out of these values again at most 2 are wrong, we get that at least 10 values correspond to polynomial $P(x)$.

But there is only one polynomial of degree 9 for any 10 values, so we conclude that polynomial $Q(x)$ must be just our polynomial $P(x)$, and we can read out correctly its coefficients which is our message.

So to summarise we used the fact that there are at most two corrupt values to first conclude that there must be at least $14-2=12$ values which correspond to a single polynomial $Q(x)$ of degree 9; we then used again the fact that there are at most two corrupt values to conclude that such polynomial $Q(x)$ shares at least $12-2=10$ values with polynomial $P(x)$ which implies that such polynomial, being of degree 9, must be identical to $P(x)$.
Let us now try sending 14 values instead.

Since at most 2 values are corrupted, we do know that there must be at least 12 values which all correspond to a polynomial of degree 9 - at least one such polynomial is just $P(x)$.

However, if any 12 values correspond to a same polynomial, say $Q(x)$, since out of these values again at most 2 are wrong, we get that at least 10 values correspond to polynomial $P(x)$.

But there is only one polynomial of degree 9 for any 10 values, so we conclude that polynomial $Q(x)$ must be just our polynomial $P(x)$, and we can read out correctly its coefficients which is our message.

So to summarise we used the fact that there are at most two corrupt values to first conclude that there must be at least $14-2=12$ values which correspond to a single polynomial $Q(x)$ of degree 9; we then used again the fact that there are at most two corrupt values to conclude that such polynomial $Q(x)$ shares at least $12-2=10$ values with polynomial $P(x)$ which implies that such polynomial, being of degree 9, must be identical to $P(x)$.
Let us now try sending 14 values instead.

Since at most 2 values are corrupted, we do know that there must be at least 12 values which all correspond to a polynomial of degree 9 - at least one such polynomial is just $P(x)$.

However, if any 12 values correspond to a same polynomial, say $Q(x)$, since out of these values again at most 2 are wrong, we get that at least 10 values correspond to polynomial $P(x)$.

But there is only one polynomial of degree 9 for any 10 values, so we conclude that polynomial $Q(x)$ must be just our polynomial $P(x)$, and we can read out correctly its coefficients which is our message.

So to summarise we used the fact that there are at most two corrupt values to first conclude that there must be at least $14-2=12$ values which correspond to a single polynomial $Q(x)$ of degree 9; we then used again the fact that there are at most two corrupt values to conclude that such polynomial $Q(x)$ shares at least $12-2=10$ values with polynomial $P(x)$ which implies that such polynomial, being of degree 9, must be identical to $P(x)$. 

Let us now try sending 14 values instead.

Since at most 2 values are corrupted, we do know that there must be at least 12 values which all correspond to a polynomial of degree 9 - at least one such polynomial is just $P(x)$.

However, if any 12 values correspond to a same polynomial, say $Q(x)$, since out of these values again at most 2 are wrong, we get that at least 10 values correspond to polynomial $P(x)$.

But there is only one polynomial of degree 9 for any 10 values, so we conclude that polynomial $Q(x)$ must be just our polynomial $P(x)$, and we can read out correctly its coefficients which is our message.

So to summarise we used the fact that there are at most two corrupt values to first conclude that there must be at least $14 - 2 = 12$ values which correspond to a single polynomial $Q(x)$ of degree 9; we then used again the fact that there are at most two corrupt values to conclude that such polynomial $Q(x)$ shares at least $12 - 2 = 10$ values with polynomial $P(x)$ which implies that such polynomial, being of degree 9, must be identical to $P(x)$.
Let us now try sending 14 values instead.

Since at most 2 values are corrupted, we do know that there must be at least 12 values which all correspond to a polynomial of degree 9 - at least one such polynomial is just $P(x)$.

However, if any 12 values correspond to a same polynomial, say $Q(x)$, since out of these values again at most 2 are wrong, we get that at least 10 values correspond to polynomial $P(x)$.

But there is only one polynomial of degree 9 for any 10 values, so we conclude that polynomial $Q(x)$ must be just our polynomial $P(x)$, and we can read out correctly its coefficients which is our message.

So to summarise we used the fact that there are at most two corrupt values to first conclude that there must be at least $14-2=12$ values which correspond to a single polynomial $Q(x)$ of degree 9; we then used again the fact that there are at most two corrupt values to conclude that such polynomial $Q(x)$ shares at least $12-2=10$ values with polynomial $P(x)$ which implies that such polynomial, being of degree 9, must be identical to $P(x)$.
In general, if we want to send a message consisting of $n$ integers, and if the channel can make at most $e$ many errors, then sending $n + 2e$ many values of a polynomial of degree $n - 1$ with coefficients being integers which we want to send, suffices to recover the message.

The reason is the same as above: if at most $e$ many symbols sent can be corrupted, then we are guaranteed that there exists a subsequence of at least $n + e$ many numbers which are values of one and the same polynomial of degree $n - 1$.

On the other hand, again because out of any sequence of $n + e$ many numbers which are the values of any single polynomial $Q(x)$ of degree $n - 1$ at most $e$ many of them are not values of our polynomial $P(x)$, we get that at least $n$ values are values of $P(x)$.

Since any two polynomials of degree $n - 1$ which share the same values for $n$ inputs must be equal, we get that $Q(x)$ must be just $P(x)$, and we can retrieve $P(x)$ correctly and read out its coefficients which form our message.
In general, if we want to send a message consisting of \( n \) integers, and if the channel can make at most \( e \) many errors, then sending \( n + 2e \) many values of a polynomial of degree \( n - 1 \) with coefficients being integers which we want to send, suffices to recover the message.

The reason is the same as above: if at most \( e \) many symbols sent can be corrupted, then we are guaranteed that there exists a subsequence of at least \( n + e \) many numbers which are values of one and the same polynomial of degree \( n - 1 \).

On the other hand, again because out of any sequence of \( n + e \) many numbers which are the values of any single polynomial \( Q(x) \) of degree \( n - 1 \) at most \( e \) many of them are not values of our polynomial \( P(x) \), we get that at least \( n \) values are values of \( P(x) \).

Since any two polynomials of degree \( n - 1 \) which share the same values for \( n \) inputs must be equal, we get that \( Q(x) \) must be just \( P(x) \), and we can retrieve \( P(x) \) correctly and read out its coefficients which form our message.
In general, if we want to send a message consisting of \( n \) integers, and if the channel can make at most \( e \) many errors, then sending \( n + 2e \) many values of a polynomial of degree \( n - 1 \) with coefficients being integers which we want to send, suffices to recover the message.

The reason is the same as above: if at most \( e \) many symbols sent can be corrupted, then we are guaranteed that there exists a subsequence of at least \( n + e \) many numbers which are values of one and the same polynomial of degree \( n - 1 \).

On the other hand, again because out of any sequence of \( n + e \) many numbers which are the values of any single polynomial \( Q(x) \) of degree \( n - 1 \) at most \( e \) many of them are not values of our polynomial \( P(x) \), we get that at least \( n \) values are values of \( P(x) \).

Since any two polynomials of degree \( n - 1 \) which share the same values for \( n \) inputs must be equal, we get that \( Q(x) \) must be just \( P(x) \), and we can retrieve \( P(x) \) correctly and read out its coefficients which form our message.
In general, if we want to send a message consisting of \(n\) integers, and if the channel can make at most \(e\) many errors, then sending \(n + 2e\) many values of a polynomial of degree \(n - 1\) with coefficients being integers which we want to send, suffices to recover the message.

The reason is the same as above: if at most \(e\) many symbols sent can be corrupted, then we are guaranteed that there exists a subsequence of at least \(n + e\) many numbers which are values of one and the same polynomial of degree \(n - 1\).

On the other hand, again because out of any sequence of \(n + e\) many numbers which are the values of any single polynomial \(Q(x)\) of degree \(n - 1\) at most \(e\) many of them are not values of our polynomial \(P(x)\), we get that at least \(n\) values are values of \(P(x)\).

Since any two polynomials of degree \(n - 1\) which share the same values for \(n\) inputs must be equal, we get that \(Q(x)\) must be just \(P(x)\), and we can retrieve \(P(x)\) correctly and read out its coefficients which form our message.
Remark: Note that, once we produce a set of \( n + e \) many values which correspond to a single polynomial of degree \( n - 1 \) we are actually guaranteed that all of these values are all correct values of \( P(x) \).

This redundancy of \( e \) many correct values beyond the \( n \) values needed to determine \( P(x) \) is the key ingredient for an efficient decoding algorithm, invented by Berlekamp and Welch, which enabled for the first time a really efficient decoding and which they patented. (This stirred a lot of legal controversy ("Can you patent a piece of mathematics?").

Before we present this algorithm, we first note that there are several problems with the above method (which is the essence of the Reed - Solomon error correction codes):

1. The values of a polynomial with integer coefficients at integer values rapidly become huge, so sending them is a huge overhead and these values are much more likely to get scrambled, having many bits;

2. If we send say 256 values out of which 28 might get scrambled, in order to retrieve a message consisting of 200 numbers, we would have to search through \( \binom{256}{228} = \binom{256}{28} > 10^{37} \) subsets of size 228 in order to find one which consists of that many values of a single polynomial of degree 199, which is obviously infeasible.
Remark: Note that, once we produce a set of $n + e$ many values which correspond to a single polynomial of degree $n - 1$ we are actually guaranteed that all of these values are all correct values of $P(x)$.

This redundancy of $e$ many correct values beyond the $n$ values needed to determine $P(x)$ is the key ingredient for an efficient decoding algorithm, invented by Berlekamp and Welch, which enabled for the first time a really efficient decoding and which they patented. (This stirred a lot of legal controversy (“Can you patent a piece of mathematics?”)).

Before we present this algorithm, we first note that there are several problems with the above method (which is the essence of the Reed - Solomon error correction codes):

1. The values of a polynomial with integer coefficients at integer values rapidly become huge, so sending them is a huge overhead and these values are much more likely to get scrambled, having many bits;

2. If we send say 256 values out of which 28 might get scrambled, in order to retrieve a message consisting of 200 numbers, we would have to search through $\binom{256}{228} = \binom{256}{28} > 10^{37}$ subsets of size 228 in order to find one which consists of that many values of a single polynomial of degree 199, which is obviously infeasible.
Remark: Note that, once we produce a set of \(n + e\) many values which correspond to a single polynomial of degree \(n - 1\) we are actually guaranteed that all of these values are all correct values of \(P(x)\).

This redundancy of \(e\) many correct values beyond the \(n\) values needed to determine \(P(x)\) is the key ingredient for an efficient decoding algorithm, invented by Berlekamp and Welch, which enabled for the first time a really efficient decoding and which they patented. (This stirred a lot of legal controversy ("Can you patent a piece of mathematics?").

Before we present this algorithm, we first note that there are several problems with the above method (which is the essence of the Reed-Solomon error correction codes):

1. The values of a polynomial with integer coefficients at integer values rapidly become huge, so sending them is a huge overhead and these values are much more likely to get scrambled, having many bits;

2. If we send say 256 values out of which 28 might get scrambled, in order to retrieve a message consisting of 200 numbers, we would have to search through \((\binom{256}{228}) = (\binom{256}{28}) > 10^{37}\) subsets of size 228 in order to find one which consists of that many values of a single polynomial of degree 199, which is obviously infeasible.
Remark: Note that, once we produce a set of $n + e$ many values which correspond to a single polynomial of degree $n - 1$ we are actually guaranteed that all of these values are all correct values of $P(x)$.

This redundancy of $e$ many correct values beyond the $n$ values needed to determine $P(x)$ is the key ingredient for an efficient decoding algorithm, invented by Berlekamp and Welch, which enabled for the first time a really efficient decoding and which they patented. (This stirred a lot of legal controversy (“Can you patent a piece of mathematics?”).

Before we present this algorithm, we first note that there are several problems with the above method (which is the essence of the Reed - Solomon error correction codes):

1. The values of a polynomial with integer coefficients at integer values rapidly become huge, so sending them is a huge overhead and these values are much more likely to get scrambled, having many bits;

2. If we send say 256 values out of which 28 might get scrambled, in order to retrieve a message consisting of 200 numbers, we would have to search through $\binom{256}{228} = \binom{256}{28} > 10^{37}$ subsets of size 228 in order to find one which consists of that many values of a single polynomial of degree 199, which is obviously infeasible.
Remark: Note that, once we produce a set of $n + e$ many values which correspond to a single polynomial of degree $n - 1$ we are actually guaranteed that all of these values are all correct values of $P(x)$.

This redundancy of $e$ many correct values beyond the $n$ values needed to determine $P(x)$ is the key ingredient for an efficient decoding algorithm, invented by Berlekamp and Welch, which enabled for the first time a really efficient decoding and which they patented. (This stirred a lot of legal controversy ("Can you patent a piece of mathematics?").

Before we present this algorithm, we first note that there are several problems with the above method (which is the essence of the Reed - Solomon error correction codes):

1. The values of a polynomial with integer coefficients at integer values rapidly become huge, so sending them is a huge overhead and these values are much more likely to get scrambled, having many bits;

2. If we send say 256 values out of which 28 might get scrambled, in order to retrieve a message consisting of 200 numbers, we would have to search through $\binom{256}{228} = \binom{256}{28} > 10^{37}$ subsets of size 228 in order to find one which consists of that many values of a single polynomial of degree 199, which is obviously infeasible.
The first problem is relatively easy to solve: instead of integers, our symbols will represent elements of a finite field, such as \( \mathbb{Z}_p \), where \( p \) is any prime number.

This is often not a convenient choice; instead, in practical applications we use finite fields of size \( p^k \) where \( p \) is prime and \( k \) is any natural number.

It can be shown that a field with \( q \) many elements exists if and only if \( q = p^k \) where \( p \) is prime and \( k \geq 1 \) any integer; moreover, for any \( q = p^k \) all fields with \( q \) many elements are isomorphic.

Such fields are called the Galois fields and (since there is only one for every \( q = p^k \)) are denoted by \( GF(p^k) \).

They can be constructed starting with the field \( \mathbb{Z}_p \) and choosing a polynomial \( P(x) \) of degree \( k \) which is irreducible over \( \mathbb{Z}_p \), i.e., a polynomial which cannot be factored as \( P(x) = Q(x)R(x) \) where neither \( Q(x) \) nor \( R(x) \) is of degree 0.

Such polynomial can be shown to always exist.

The elements of the field are polynomials of degree at most \( k - 1 \); the sum and product of two such polynomials is just the reminder of the usual sum and product of these two polynomials when divided by the irreducible polynomial \( P(x) \).
The first problem is relatively easy to solve: instead of integers, our symbols will represent elements of a finite field, such as \( \mathbb{Z}_p \), where \( p \) is any prime number.

This is often not a convenient choice; instead, in practical applications we use finite fields of size \( p^k \) where \( p \) is prime and \( k \) is any natural number.

It can be shown that a field with \( q \) many elements exists if and only if \( q = p^k \) where \( p \) is prime and \( k \geq 1 \) any integer; moreover, for any \( q = p^k \) all fields with \( q \) many elements are isomorphic.

Such fields are called the Galois fields and (since there is only one for every \( q = p^k \)) are denoted by \( GF(p^k) \).

They can be constructed starting with the field \( \mathbb{Z}_p \) and choosing a polynomial \( P(x) \) of degree \( k \) which is irreducible over \( \mathbb{Z}_p \), i.e., a polynomial which cannot be factored as \( P(x) = Q(x)R(x) \) where neither \( Q(x) \) nor \( R(x) \) is of degree 0.

Such polynomial can be shown to always exist.

The elements of the field are polynomials of degree at most \( k - 1 \); the sum and product of two such polynomials is just the reminder of the usual sum and product of these two polynomials when divided by the irreducible polynomial \( P(x) \).
The first problem is relatively easy to solve: instead of integers, our symbols will represent elements of a finite field, such as \( \mathbb{Z}_p \), where \( p \) is any prime number.

This is often not a convenient choice; instead, in practical applications we use finite fields of size \( p^k \) where \( p \) is prime and \( k \) is any natural number.

It can be shown that a field with \( q \) many elements exists if and only if \( q = p^k \) where \( p \) is prime and \( k \geq 1 \) any integer; moreover, for any \( q = p^k \) all fields with \( q \) many elements are isomorphic.

Such fields are called the Galois fields and (since there is only one for every \( q = p^k \)) are denoted by \( GF(p^k) \).

They can be constructed starting with the field \( \mathbb{Z}_p \) and choosing a polynomial \( P(x) \) of degree \( k \) which is irreducible over \( \mathbb{Z}_p \), i.e., a polynomial which cannot be factored as \( P(x) = Q(x)R(x) \) where neither \( Q(x) \) nor \( R(x) \) is of degree 0.

Such polynomial can be shown to always exist.

The elements of the field are polynomials of degree at most \( k - 1 \); the sum and product of two such polynomials is just the reminder of the usual sum and product of these two polynomials when divided by the irreducible polynomial \( P(x) \).
The first problem is relatively easy to solve: instead of integers, our symbols will represent elements of a finite field, such as \( \mathbb{Z}_p \), where \( p \) is any prime number.

This is often not a convenient choice; instead, in practical applications we use finite fields of size \( p^k \) where \( p \) is prime and \( k \) is any natural number.

It can be shown that a field with \( q \) many elements exists if and only if \( q = p^k \) where \( p \) is prime and \( k \geq 1 \) any integer; moreover, for any \( q = p^k \) all fields with \( q \) many elements are isomorphic.

Such fields are called the Galois fields and (since there is only one for every \( q = p^k \)) are denoted by \( GF(p^k) \).

They can be constructed starting with the field \( \mathbb{Z}_p \) and choosing a polynomial \( P(x) \) of degree \( k \) which is irreducible over \( \mathbb{Z}_p \), i.e., a polynomial which cannot be factored as \( P(x) = Q(x)R(x) \) where neither \( Q(x) \) nor \( R(x) \) is of degree 0.

Such polynomial can be shown to always exist.

The elements of the field are polynomials of degree at most \( k - 1 \); the sum and product of two such polynomials is just the reminder of the usual sum and product of these two polynomials when divided by the irreducible polynomial \( P(x) \).
The first problem is relatively easy to solve: instead of integers, our symbols will represent elements of a finite field, such as \( \mathbb{Z}_p \), where \( p \) is any prime number.

This is often not a convenient choice; instead, in practical applications we use finite fields of size \( p^k \) where \( p \) is prime and \( k \) is any natural number.

It can be shown that a field with \( q \) many elements exists if and only if \( q = p^k \) where \( p \) is prime and \( k \geq 1 \) any integer; moreover, for any \( q = p^k \) all fields with \( q \) many elements are isomorphic.

Such fields are called the Galois fields and (since there is only one for every \( q = p^k \)) are denoted by \( GF(p^k) \).

They can be constructed starting with the field \( \mathbb{Z}_p \) and choosing a polynomial \( P(x) \) of degree \( k \) which is irreducible over \( \mathbb{Z}_p \), i.e., a polynomial which cannot be factored as \( P(x) = Q(x)R(x) \) where neither \( Q(x) \) nor \( R(x) \) is of degree 0.

Such polynomial can be shown to always exist.

The elements of the field are polynomials of degree at most \( k - 1 \); the sum and product of two such polynomials is just the reminder of the usual sum and product of these two polynomials when divided by the irreducible polynomial \( P(x) \).
The first problem is relatively easy to solve: instead of integers, our symbols will represent elements of a finite field, such as $\mathbb{Z}_p$, where $p$ is any prime number.

This is often not a convenient choice; instead, in practical applications we use finite fields of size $p^k$ where $p$ is prime and $k$ is any natural number.

It can be shown that a field with $q$ many elements exists if and only if $q = p^k$ where $p$ is prime and $k \geq 1$ any integer; moreover, for any $q = p^k$ all fields with $q$ many elements are isomorphic.

Such fields are called the Galois fields and (since there is only one for every $q = p^k$) are denoted by $GF(p^k)$.

They can be constructed starting with the field $\mathbb{Z}_p$ and choosing a polynomial $P(x)$ of degree $k$ which is irreducible over $\mathbb{Z}_p$, i.e., a polynomial which cannot be factored as $P(x) = Q(x)R(x)$ where neither $Q(x)$ nor $R(x)$ is of degree 0.

Such polynomial can be shown to always exist.

The elements of the field are polynomials of degree at most $k - 1$; the sum and product of two such polynomials is just the reminder of the usual sum and product of these two polynomials when divided by the irreducible polynomial $P(x)$. 
In practice $GF(2^8) = GF(256)$ is most frequently used because elements of such a field can be represented by a single byte. Multiplication and addition of these elements are precomputed and stored in the corresponding tables.

This makes evaluation of any polynomial with coefficients from $GF(p^k)$ at any element of such a field very efficient and of course the values of such polynomials stay in the same field.

The second problem mentioned above was solved by Berlekamp and Welch. To keep the notation simple, rather than working in a Galois field, we will work in $\mathbb{Z}_p$ because the operations in such a field are just the usual modular addition and multiplication; however, everything we do holds in any finite field, with the appropriately defined addition and multiplication operations.

The main trick is to introduce a polynomial $E(x)$ of degree at most $e$ (the max number of errors) which will mask the errors by satisfying $E(i) = 0$ whenever $P(i) \neq b_i$ and by introducing a polynomial $Q(x) = P(x)E(x)$ which will also vanish at these $i$ and which will satisfy $Q(i) = b_iE(i)$ for all $0 \leq i \leq n - 1$. 
In practice $GF(2^8) = GF(256)$ is most frequently used because elements of such a field can be represented by a single byte. Multiplication and addition of these elements are precomputed and stored in the corresponding tables.

This makes evaluation of any polynomial with coefficients from $GF(p^k)$ at any element of such a field very efficient and of course the values of such polynomials stay in the same field.

The second problem mentioned above was solved by Berlekamp and Welch. To keep the notation simple, rather than working in a Galois field, we will work in $\mathbb{Z}_p$ because the operations in such a field are just the usual modular addition and multiplication; however, everything we do holds in any finite field, with the appropriately defined addition and multiplication operations.

The main trick is to introduce a polynomial $E(x)$ of degree at most $e$ (the max number of errors) which will mask the errors by satisfying $E(i) = 0$ whenever $P(i) \neq b_i$ and by introducing a polynomial $Q(x) = P(x)E(x)$ which will also vanish at these $i$ and which will satisfy $Q(i) = b_iE(i)$ for all $0 \leq i \leq n - 1$. 
In practice $GF(2^8) = GF(256)$ is most frequently used because elements of such a field can be represented by a single byte. Multiplication and addition of these elements are precomputed and stored in the corresponding tables.

This makes evaluation of any polynomial with coefficients from $GF(p^k)$ at any element of such a field very efficient and of course the values of such polynomials stay in the same field.

The second problem mentioned above was solved by Berlekamp and Welch. To keep the notation simple, rather than working in a Galois field, we will work in $\mathbb{Z}_p$ because the operations in such a field are just the usual modular addition and multiplication; however, everything we do holds in any finite field, with the appropriately defined addition and multiplication operations.

The main trick is to introduce a polynomial $E(x)$ of degree at most $e$ (the max number of errors) which will mask the errors by satisfying $E(i) = 0$ whenever $P(i) \neq b_i$ and by introducing a polynomial $Q(x) = P(x)E(x)$ which will also vanish at these $i$ and which will satisfy $Q(i) = b_i E(i)$ for all $0 \leq i \leq n - 1$. 
• In practice $GF(2^8) = GF(256)$ is most frequently used because elements of such a field can be represented by a single byte. Multiplication and addition of these elements are precomputed and stored in the corresponding tables.

• This makes evaluation of any polynomial with coefficients from $GF(p^k)$ at any element of such a field very efficient and of course the values of such polynomials stay in the same field.

• The second problem mentioned above was solved by Berlekamp and Welch. To keep the notation simple, rather than working in a Galois field, we will work in $\mathbb{Z}_p$ because the operations in such a field are just the usual modular addition and multiplication; however, everything we do holds in any finite field, with the appropriately defined addition and multiplication operations.

• The main trick is to introduce a polynomial $E(x)$ of degree at most $e$ (the max number of errors) which will mask the errors by satisfying $E(i) = 0$ whenever $P(i) \neq b_i$ and by introducing a polynomial $Q(x) = P(x)E(x)$ which will also vanish at these $i$ and which will satisfy $Q(i) = b_i E(i)$ for all $0 \leq i \leq n - 1$. 
Lemma

Let $P(x)$ be a polynomial of degree $m - 1$, and let the transmitted code vector be 
\[ \tau = \langle P(0), P(1), \ldots, P(n-2), P(n-1) \rangle, \]
and let the received vector be 
\[ b = \langle b_0, b_1, \ldots, b_{n-2}, b_{n-1} \rangle; \]
assume also that there are at most $e$ many errors, i.e., at most $e$ values $i$ for which $b_i \neq P(i)$. Then there exist two non zero polynomials, $E(x)$ of degree at most $e$ and $Q(x)$ of degree at most $m - 1 + e$ such that
\[ Q(i) = b_i E(i) \quad \text{for all} \quad 0 \leq i \leq n - 1 \]  
(5)

- Let the number of errors be $k \leq e$, and let $\mathcal{E} = \{i : b_i \neq P(i)\}$.
- We set $E(x) = \prod_{i \in \mathcal{E}} (x - i)$ and $Q(x) = P(x)E(x)$.
- The degree of $E(x)$ is at most $e$ and since the degree of $P(x)$ is $m - 1$, the degree of $Q(x) = P(x)E(x)$ is at most $m - 1 + e$.
- We now show that $Q(i) = b_i E(i)$ holds for all $0 \leq i \leq n - 1$.
- We distinguish two cases.
  - If $i \in \mathcal{E}$, then $E(i) = 0$ and thus also both $Q(i) = P(i)E(i) = 0$ and $b_i E(i) = 0$; thus, $Q(i) = b_i E(i)$ holds.
  - If $i \not\in \mathcal{E}$, then $b_i = P(i)$ and consequently also $b_i E(i) = P(i)E(i) = Q(i)$.

Thus, for all $0 \leq i \leq n - 1$ we have $Q(i) = b_i E(i)$. 
Lemma

Let $P(x)$ be a polynomial of degree $m - 1$, and let the transmitted code vector be $\tau = \langle P(0), P(1), \ldots, P(n - 2), P(n - 1) \rangle$, and let the received vector be $b = \langle b_0, b_1, \ldots, b_{n-2}, b_{n-1} \rangle$; assume also that there are at most $e$ many errors, i.e., at most $e$ values $i$ for which $b_i \neq P(i)$. Then there exist two non zero polynomials, $E(x)$ of degree at most $e$ and $Q(x)$ of degree at most $m - 1 + e$ such that

$$Q(i) = b_i E(i) \quad \text{for all} \quad 0 \leq i \leq n - 1 \quad (5)$$

- Let the number of errors be $k \leq e$, and let $\mathcal{E} = \{i : b_i \neq P(i)\}$.
- We set $E(x) = \prod_{i \in \mathcal{E}} (x - i)$ and $Q(x) = P(x)E(x)$.
- The degree of $E(x)$ is at most $e$ and since the degree of $P(x)$ is $m - 1$, the degree of $Q(x) = P(x)E(x)$ is at most $m - 1 + e$.
- We now show that $Q(i) = b_i E(i)$ holds for all $0 \leq i \leq n - 1$.
- We distinguish two cases.
  - If $i \in \mathcal{E}$, then $E(i) = 0$ and thus also both $Q(i) = P(i) E(i) = 0$ and $b_i E(i) = 0$; thus, $Q(i) = b_i E(i)$ holds.
  - If $i \notin \mathcal{E}$, then $b_i = P(i)$ and consequently also $b_i E(i) = P(i) E(i) = Q(i)$.

Thus, for all $0 \leq i \leq n - 1$ we have $Q(i) = b_i E(i)$. 

COMP4121
Lemma

Let $P(x)$ be a polynomial of degree $m - 1$, and let the transmitted code vector be $	au = \langle P(0), P(1), \ldots, P(n - 2), P(n - 1) \rangle$, and let the received vector be $b = \langle b_0, b_1, \ldots, b_{n-2}, b_{n-1} \rangle$; assume also that there are at most $e$ many errors, i.e., at most $e$ values $i$ for which $b_i \neq P(i)$. Then there exist two non zero polynomials, $E(x)$ of degree at most $e$ and $Q(x)$ of degree at most $m - 1 + e$ such that

$$Q(i) = b_i E(i) \quad \text{for all} \quad 0 \leq i \leq n - 1$$

Let the number of errors be $k \leq e$, and let $\mathcal{E} = \{i : b_i \neq P(i)\}$.

We set $E(x) = \prod_{i \in \mathcal{E}} (x - i)$ and $Q(x) = P(x)E(x)$.

The degree of $E(x)$ is at most $e$ and since the degree of $P(x)$ is $m - 1$, the degree of $Q(x) = P(x)E(x)$ is at most $m - 1 + e$.

We now show that $Q(i) = b_i E(i)$ holds for all $0 \leq i \leq n - 1$.

We distinguish two cases.

- If $i \in \mathcal{E}$, then $E(i) = 0$ and thus also both $Q(i) = P(i)E(i) = 0$ and $b_i E(i) = 0$; thus, $Q(i) = b_i E(i)$ holds.
- If $i \notin \mathcal{E}$, then $b_i = P(i)$ and consequently also $b_i E(i) = P(i)E(i) = Q(i)$.

Thus, for all $0 \leq i \leq n - 1$ we have $Q(i) = b_i E(i)$. 
Lemma

Let $P(x)$ be a polynomial of degree $m - 1$, and let the transmitted code vector be
$
\tau = \langle P(0), P(1), \ldots, P(n - 2), P(n - 1) \rangle,$
and let the received vector be
$b = \langle b_0, b_1, \ldots, b_{n-2}, b_{n-1} \rangle$; assume also that there are at most $e$ many errors, i.e.,
at most $e$ values $i$ for which $b_i \neq P(i)$. Then there exist two non zero polynomials,
$E(x)$ of degree at most $e$ and $Q(x)$ of degree at most $m - 1 + e$ such that

$$Q(i) = b_i E(i) \quad \text{for all} \quad 0 \leq i \leq n - 1 \quad (5)$$

- Let the number of errors be $k \leq e$, and let $E = \{i : b_i \neq P(i)\}$.
- We set $E(x) = \prod_{i \in E} (x - i)$ and $Q(x) = P(x)E(x)$.
- The degree of $E(x)$ is at most $e$ and since the degree of $P(x)$ is $m - 1$, the
degree of $Q(x) = P(x)E(x)$ is at most $m - 1 + e$.
- We now show that $Q(i) = b_i E(i)$ holds for all $0 \leq i \leq n - 1$.
- We distinguish two cases.
  - If $i \in E$, then $E(i) = 0$ and thus also both $Q(i) = P(i)E(i) = 0$ and
    $b_i E(i) = 0$; thus, $Q(i) = b_i E(i)$ holds.
  - If $i \notin E$, then $b_i = P(i)$ and consequently also
    $b_i E(i) = P(i)E(i) = Q(i)$.

Thus, for all $0 \leq i \leq n - 1$ we have $Q(i) = b_i E(i)$. 
Lemma

Let $P(x)$ be a polynomial of degree $m - 1$, and let the transmitted code vector be $\tau = \langle P(0), P(1), \ldots, P(n - 2), P(n - 1) \rangle$, and let the received vector be $b = \langle b_0, b_1, \ldots, b_{n-2}, b_{n-1} \rangle$; assume also that there are at most $e$ many errors, i.e., at most $e$ values $i$ for which $b_i \neq P(i)$. Then there exist two non zero polynomials, $E(x)$ of degree at most $e$ and $Q(x)$ of degree at most $m - 1 + e$ such that

$$Q(i) = b_i E(i) \quad \text{for all} \quad 0 \leq i \leq n - 1 \quad (5)$$

- Let the number of errors be $k \leq e$, and let $\mathcal{E} = \{ i : b_i \neq P(i) \}$.
- We set $E(x) = \prod_{i \in \mathcal{E}} (x - i)$ and $Q(x) = P(x)E(x)$.
- The degree of $E(x)$ is at most $e$ and since the degree of $P(x)$ is $m - 1$, the degree of $Q(x) = P(x)E(x)$ is at most $m - 1 + e$.
- We now show that $Q(i) = b_i E(i)$ holds for all $0 \leq i \leq n - 1$.

  - We distinguish two cases.
    - If $i \in \mathcal{E}$, then $E(i) = 0$ and thus also both $Q(i) = P(i)E(i) = 0$ and $b_i E(i) = 0$; thus, $Q(i) = b_i E(i)$ holds.
    - If $i \notin \mathcal{E}$, then $b_i = P(i)$ and consequently also $b_i E(i) = P(i)E(i) = Q(i)$.
    
    Thus, for all $0 \leq i \leq n - 1$ we have $Q(i) = b_i E(i)$. 

Lemma

Let $P(x)$ be a polynomial of degree $m - 1$, and let the transmitted code vector be
\[ \tau = \langle P(0), P(1), \ldots, P(n - 2), P(n - 1) \rangle, \]
and let the received vector be
\[ b = \langle b_0, b_1, \ldots, b_{n-2}, b_{n-1} \rangle; \]
assume also that there are at most $e$ many errors, i.e., at most $e$ values $i$ for which $b_i \neq P(i)$. Then there exist two non zero polynomials, $E(x)$ of degree at most $e$ and $Q(x)$ of degree at most $m - 1 + e$ such that

\[ Q(i) = b_i E(i) \quad \text{for all} \quad 0 \leq i \leq n - 1 \]  

(5)

- Let the number of errors be $k \leq e$, and let $E = \{i : b_i \neq P(i)\}$.
- We set $E(x) = \prod_{i \in E} (x - i)$ and $Q(x) = P(x)E(x)$.
- The degree of $E(x)$ is at most $e$ and since the degree of $P(x)$ is $m - 1$, the degree of $Q(x) = P(x)E(x)$ is at most $m - 1 + e$.
- We now show that $Q(i) = b_i E(i)$ holds for all $0 \leq i \leq n - 1$.
- We distinguish two cases.
  - If $i \in E$, then $E(i) = 0$ and thus also both $Q(i) = P(i)E(i) = 0$ and $b_i E(i) = 0$; thus, $Q(i) = b_i E(i)$ holds.
  - If $i \notin E$, then $b_i = P(i)$ and consequently also $b_i E(i) = P(i)E(i) = Q(i)$.

Thus, for all $0 \leq i \leq n - 1$ we have $Q(i) = b_i E(i)$. 

Lemma

Let \( P(x) \) be a polynomial of degree \( m - 1 \), and let the transmitted code vector be \( \tau = \langle P(0), P(1), \ldots, P(n - 2), P(n - 1) \rangle \), and let the received vector be \( \mathbf{b} = \langle b_0, b_1, \ldots, b_{n-2}, b_{n-1} \rangle \); assume also that there are at most \( e \) many errors, i.e., at most \( e \) values \( i \) for which \( b_i \neq P(i) \). Then there exist two non zero polynomials, \( E(x) \) of degree at most \( e \) and \( Q(x) \) of degree at most \( m - 1 + e \) such that

\[
Q(i) = b_i E(i) \quad \text{for all} \quad 0 \leq i \leq n - 1
\]  

(5)

- Let the number of errors be \( k \leq e \), and let \( \mathcal{E} = \{i : b_i \neq P(i)\} \).
- We set \( E(x) = \prod_{i \in \mathcal{E}} (x - i) \) and \( Q(x) = P(x)E(x) \).
- The degree of \( E(x) \) is at most \( e \) and since the degree of \( P(x) \) is \( m - 1 \), the degree of \( Q(x) = P(x)E(x) \) is at most \( m - 1 + e \).
- We now show that \( Q(i) = b_i E(i) \) holds for all \( 0 \leq i \leq n - 1 \).
- We distinguish two cases.
  - If \( i \in \mathcal{E} \), then \( E(i) = 0 \) and thus also both \( Q(i) = P(i)E(i) = 0 \) and \( b_i E(i) = 0 \); thus, \( Q(i) = b_i E(i) \) holds.
  - If \( i \notin \mathcal{E} \), then \( b_i = P(i) \) and consequently also \( b_i E(i) = P(i)E(i) = Q(i) \).

Thus, for all \( 0 \leq i \leq n - 1 \) we have \( Q(i) = b_i E(i) \).
Lemma

Let $P(x)$ be a polynomial of degree $m - 1$, and let the transmitted code vector be $\tau = \langle P(0), P(1), \ldots, P(n-2), P(n-1) \rangle$, and let the received vector be $b = \langle b_0, b_1, \ldots, b_{n-2}, b_{n-1} \rangle$; assume also that there are at most $e$ many errors, i.e., at most $e$ values $i$ for which $b_i \neq P(i)$. Then there exist two non zero polynomials, $E(x)$ of degree at most $e$ and $Q(x)$ of degree at most $m - 1 + e$ such that

$$Q(i) = b_i E(i) \quad \text{for all} \quad 0 \leq i \leq n - 1 \quad (5)$$

- Let the number of errors be $k \leq e$, and let $E = \{i : b_i \neq P(i)\}$.
- We set $E(x) = \prod_{i \in E} (x - i)$ and $Q(x) = P(x)E(x)$.
- The degree of $E(x)$ is at most $e$ and since the degree of $P(x)$ is $m - 1$, the degree of $Q(x) = P(x)E(x)$ is at most $m - 1 + e$.
- We now show that $Q(i) = b_i E(i)$ holds for all $0 \leq i \leq n - 1$.
- We distinguish two cases.
  - If $i \in E$, then $E(i) = 0$ and thus also both $Q(i) = P(i) E(i) = 0$ and $b_i E(i) = 0$; thus, $Q(i) = b_i E(i)$ holds.
  - If $i \not\in E$, then $b_i = P(i)$ and consequently also $b_i E(i) = P(i) E(i) = Q(i)$.

Thus, for all $0 \leq i \leq n - 1$ we have $Q(i) = b_i E(i)$. 
Lemma

Let $P(x)$ be a polynomial of degree $m - 1$, and let the transmitted code vector be $\tau = \langle P(0), P(1), \ldots, P(n - 2), P(n - 1) \rangle$, and let the received vector be $b = \langle b_0, b_1, \ldots, b_{n-2}, b_{n-1} \rangle$; assume also that there are at most $e$ many errors, i.e., at most $e$ values $i$ for which $b_i \neq P(i)$. Then there exist two non zero polynomials, $E(x)$ of degree at most $e$ and $Q(x)$ of degree at most $m - 1 + e$ such that

$$Q(i) = b_i E(i) \quad \text{for all} \quad 0 \leq i \leq n - 1 \quad (5)$$

- Let the number of errors be $k \leq e$, and let $\mathcal{E} = \{i : b_i \neq P(i)\}$.
- We set $E(x) = \prod_{i \in \mathcal{E}} (x - i)$ and $Q(x) = P(x)E(x)$.
- The degree of $E(x)$ is at most $e$ and since the degree of $P(x)$ is $m - 1$, the degree of $Q(x) = P(x)E(x)$ is at most $m - 1 + e$.
- We now show that $Q(i) = b_i E(i)$ holds for all $0 \leq i \leq n - 1$.
- We distinguish two cases.
  - If $i \in \mathcal{E}$, then $E(i) = 0$ and thus also both $Q(i) = P(i)E(i) = 0$ and $b_i E(i) = 0$; thus, $Q(i) = b_i E(i)$ holds.
  - If $i \not\in \mathcal{E}$, then $b_i = P(i)$ and consequently also $b_i E(i) = P(i)E(i) = Q(i)$.

Thus, for all $0 \leq i \leq n - 1$ we have $Q(i) = b_i E(i)$. 

Lemma

If the number of errors is at most $e$, then for any two polynomials $E(x)$ and $Q(x)$, such that $E(x)$ is of degree at most $e$ and $Q(x)$ is of degree at most $m - 1 + e$ and which satisfy $Q(i) = b_i E(i)$ for all $i$ such that $0 \leq i \leq n - 1$ we have $Q(x) = P(x)E(x)$ and thus we can obtain $P(x)$ as $P(x) = Q(x)/E(x)$.

- **Proof:** Since $P(i) = b_i$ for at least $n - e = m + e$ many values, we get that also $P(i)E(i) = b_iE(i)$ for at least $m + e$ many values.
- Since for all $0 \leq i \leq n - 1$ we have $Q(i) = b_iE(i)$ we obtain that for at least $m + e$ many values $Q(i) = P(i)E(i)$.
- However, both $Q(x)$ and $P(x)E(x)$ are polynomials of degree at most $m - 1 + e$, so we conclude that they must coincide. i.e., $Q(x) = P(x)E(x)$.
Lemma

If the number of errors is at most $e$, then for any two polynomials $E(x)$ and $Q(x)$, such that $E(x)$ is of degree at most $e$ and $Q(x)$ is of degree at most $m - 1 + e$ and which satisfy $Q(i) = b_i E(i)$ for all $i$ such that $0 \leq i \leq n - 1$ we have $Q(x) = P(x) E(x)$ and thus we can obtain $P(x)$ as $P(x) = Q(x) / E(x)$.

- **Proof:** Since $P(i) = b_i$ for at least $n - e = m + e$ many values, we get that also $P(i) E(i) = b_i E(i)$ for at least $m + e$ many values.
- Since for all $0 \leq i \leq n - 1$ we have $Q(i) = b_i E(i)$ we obtain that for at least $m + e$ many values $Q(i) = P(i) E(i)$.
- However, both $Q(x)$ and $P(x) E(x)$ are polynomials of degree at most $m - 1 + e$, so we conclude that they must coincide. i.e., $Q(x) = P(x) E(x)$.
Lemma

If the number of errors is at most \( e \), then for any two polynomials \( E(x) \) and \( Q(x) \), such that \( E(x) \) is of degree at most \( e \) and \( Q(x) \) is of degree at most \( m - 1 + e \) and which satisfy \( Q(i) = b_i E(i) \) for all \( i \) such that \( 0 \leq i \leq n - 1 \) we have \( Q(x) = P(x)E(x) \) and thus we can obtain \( P(x) \) as \( P(x) = Q(x)/E(x) \).

Proof: Since \( P(i) = b_i \) for at least \( n - e = m + e \) many values, we get that also \( P(i)E(i) = b_i E(i) \) for at least \( m + e \) many values.

Since for all \( 0 \leq i \leq n - 1 \) we have \( Q(i) = b_i E(i) \) we obtain that for at least \( m + e \) many values \( Q(i) = P(i)E(i) \).

However, both \( Q(x) \) and \( P(x)E(x) \) are polynomials of degree at most \( m - 1 + e \), so we conclude that they must coincide. i.e., \( Q(x) = P(x)E(x) \).
Lemma

If the number of errors is at most $e$, then for any two polynomials $E(x)$ and $Q(x)$, such that $E(x)$ is of degree at most $e$ and $Q(x)$ is of degree at most $m - 1 + e$ and which satisfy $Q(i) = b_i E(i)$ for all $i$ such that $0 \leq i \leq n - 1$ we have $Q(x) = P(x)E(x)$ and thus we can obtain $P(x)$ as $P(x) = Q(x)/E(x)$.

Proof: Since $P(i) = b_i$ for at least $n - e = m + e$ many values, we get that also $P(i)E(i) = b_i E(i)$ for at least $m + e$ many values.

Since for all $0 \leq i \leq n - 1$ we have $Q(i) = b_i E(i)$ we obtain that for at least $m + e$ many values $Q(i) = P(i)E(i)$.

However, both $Q(x)$ and $P(x)E(x)$ are polynomials of degree at most $m - 1 + e$, so we conclude that they must coincide. i.e., $Q(x) = P(x)E(x)$. 

The above two lemmata allow for an efficient decoding algorithm. We represent $Q(x)$ and $E(x)$ in the coefficient form as

$$Q(x) = \sum_{i=0}^{m-1+e} u_i x^i; \quad E(x) = \sum_{j=0}^{e} v_j x^j,$$

where $u_i, v_j$ are variables.

We then substitute these expressions into equations $Q(i) = b_i E(i)$ for all $0 \leq i \leq n - 1$ which produces $n$ equations linear in $m + e + e + 1 = n + 1$ variables $u_i, v_j$.

Lemma 1 guarantees that there exists a non trivial (i.e., non zero) solution of such a linear system; Lemma 2 guarantees that any non trivial solution of these equations produces values for $u_i, v_j$ such that the corresponding polynomials $Q(x)$ and $E(x)$ satisfy $Q(x)/E(x) = P(x)$.

If the number of actual errors is less than $e$ the above system will be underdetermined but this does not matter because we have shown that any $Q(x)$ and $E(x)$ will produce the same polynomial $P(x)$!

Fast error correction algorithms revolutionised digital telecommunications, allowing the transceivers to operate near theoretical channel capacity by enabling them to be error tolerant.
The above two lemmata allow for an efficient decoding algorithm. We represent $Q(x)$ and $E(x)$ in the coefficient form as

$$Q(x) = \sum_{i=0}^{m-1+e} u_i x^i; \quad E(x) = \sum_{j=0}^{e} v_j x^j,$$

where $u_i, v_j$ are variables.

We then substitute these expressions into equations $Q(i) = b_i E(i)$ for all $0 \leq i \leq n - 1$ which produces $n$ equations linear in $m + e + e + 1 = n + 1$ variables $u_i, v_j$.

Lemma 1 guarantees that there exists a non trivial (i.e., non zero) solution of such a linear system; Lemma 2 guarantees that any non trivial solution of these equations produces values for $u_i, v_j$ such that the corresponding polynomials $Q(x)$ and $E(x)$ satisfy $Q(x)/E(x) = P(x)$.

If the number of actual errors is less than $e$ the above system will be underdetermined but this does not matter because we have shown that any $Q(x)$ and $E(x)$ will produce the same polynomial $P(x)$!

Fast error correction algorithms revolutionised digital telecommunications, allowing the transceivers to operate near theoretical channel capacity by enabling them to be error tolerant.
The above two lemmata allow for an efficient decoding algorithm. We represent \( Q(x) \) and \( E(x) \) in the coefficient form as

\[
Q(x) = \sum_{i=0}^{m-1+e} u_i x^i; \quad E(x) = \sum_{j=0}^{e} v_j x^j,
\]

where \( u_i, v_j \) are variables.

We then substitute these expressions into equations \( Q(i) = b_i E(i) \) for all \( 0 \leq i \leq n - 1 \) which produces \( n \) equations linear in \( m + e + e + 1 = n + 1 \) variables \( u_i, v_j \).

Lemma 1 guarantees that there exists a non trivial (i.e., non zero) solution of such a linear system; Lemma 2 guarantees that any non trivial solution of these equations produces values for \( u_i, v_j \) such that the corresponding polynomials \( Q(x) \) and \( E(x) \) satisfy \( Q(x)/E(x) = P(x) \).

If the number of actual errors is less than \( e \) the above system will be underdetermined but this does not matter because we have shown that any \( Q(x) \) and \( E(x) \) will produce the same polynomial \( P(x) \)!

Fast error correction algorithms revolutionised digital telecommunications, allowing the transceivers to operate near theoretical channel capacity by enabling them to be error tolerant.
The above two lemmata allow for an efficient decoding algorithm. We represent $Q(x)$ and $E(x)$ in the coefficient form as

$$Q(x) = \sum_{i=0}^{m-1+e} u_i x^i; \quad E(x) = \sum_{j=0}^{e} v_j x^j,$$

where $u_i, v_j$ are variables.

We then substitute these expressions into equations $Q(i) = b_i E(i)$ for all $0 \leq i \leq n - 1$ which produces $n$ equations linear in $m + e + e + 1 = n + 1$ variables $u_i, v_j$.

Lemma 1 guarantees that there exists a non trivial (i.e., non zero) solution of such a linear system; Lemma 2 guarantees that any non trivial solution of these equations produces values for $u_i, v_j$ such that the corresponding polynomials $Q(x)$ and $E(x)$ satisfy $Q(x)/E(x) = P(x)$.

If the number of actual errors is less than $e$ the above system will be underdetermined but this does not matter because we have shown that any $Q(x)$ and $E(x)$ will produce the same polynomial $P(x)$!

Fast error correction algorithms revolutionised digital telecommunications, allowing the transceivers to operate near theoretical channel capacity by enabling them to be error tolerant.
The above two lemmata allow for an efficient decoding algorithm. We represent $Q(x)$ and $E(x)$ in the coefficient form as

$$Q(x) = \sum_{i=0}^{m-1+e} u_i x^i; \quad E(x) = \sum_{j=0}^{e} v_j x^j,$$

where $u_i, v_j$ are variables.

We then substitute these expressions into equations $Q(i) = b_i E(i)$ for all $0 \leq i \leq n - 1$ which produces $n$ equations linear in $m + e + e + 1 = n + 1$ variables $u_i, v_j$.

Lemma 1 guarantees that there exists a non trivial (i.e., non zero) solution of such a linear system; Lemma 2 guarantees that any non trivial solution of these equations produces values for $u_i, v_j$ such that the corresponding polynomials $Q(x)$ and $E(x)$ satisfy $Q(x)/E(x) = P(x)$.

If the number of actual errors is less than $e$ the above system will be underdetermined but this does not matter because we have shown that any $Q(x)$ and $E(x)$ will produce the same polynomial $P(x)$!

Fast error correction algorithms revolutionised digital telecommunications, allowing the transceivers to operate near theoretical channel capacity by enabling them to be error tolerant.