Теорія інформації та кодування
Ч-1

1. **Введення** 3
1.1. Модель радіотехнічної системи передачі інформації 4
1.2. Джерело інформації 9
1.3. Теорема дискретизації 12
1.4. Кількість інформації, ентропія джерела повідомлень 12
1.6.1. Ентропія складних повідомлень, надмірність джерела 15
2. Основи економного кодування 20
2.1. Мета стискування даних і типи систем стискування 26
2.1.1. Стискування без втрат інформації 27
2.1.2. Стискування з втратою інформації 27
2.2. Коди без пам'яті. Коди Хаффмена 32
2.2.1. Алгоритм Хаффмена 33
2.2.2. Кордони ентропії для коди Хаффмена 34
2.3. Коди з пам'яттю 36
2.4. Арифметичне кодування 38
2.4.1. Кодування 39
2.4.2. Декодування 40
2.5. Словарні методи кодування. Метод Зіва-Лемпела 42
2.5.1. Кодування 45
2.5.2. Декодування 47
2.6. Кодування довжин повторень 48
2.7. Диференціальне кодування 50
2.8. Методи стискування з втратою інформації 51
2.8.1. Кодування перетворень. Стандарт стискування JPEG 52
2.8.2. Фрактальний метод 61
2.8.3. Рекурсивний (хвилевий) алгоритм 63
2.9. Методи стискування рухливих зображень (відео) 65
2.10. Методи стискування мовних сигналів 68
Література 73

1. Введення

Здобуття, передача, обробка і зберігання інформації - це що найдинамічніше розвиваються в останні десятиліття і перспективні області людської діяльності. Десятки тисяч крупних, середніх і дрібних фірм у всьому світі, з річним зворотом в сотні мільярдів доларів, займаються дослідженням, розробкою, виробництвом, продажем і експлуатацією всіляких систем і пристроїв передачі інформації.

Розібратися в існуючому різноманітті методів передачі, а також різних систем, призначених для передачі інформації, зрозуміти загальні принципи їх побудови і роботи, вивчити сучасні підходи до розробки систем передачі інформації допоможе курс "Теорія інформації та кодування ".

Спершу визначимо предмет вивчення.

Як випливає з самої назви, радіотехнічні системи передачі інформації (РТС ПІ) призначені для передачі не яких-небудь об'єктів, енергії або сигналів, а ІНФОРМАЦІЇ. Електромагнітні хвилі (сигнали) в цих системах використовуються лише як переносники інформації від джерела до її споживача. У цьому полягає їх головна відмінність від безлічі інших радіотехнічних систем, також здійснюючу передачу і прийом радіосигналів. До РТС ПІ, або радіотехнічні системи зв'язного типа, як їх інакше називають, відносяться системи радіозв'язку (у тому числі системи стільникового радіотелефонного зв'язку), системи радіомовлення і телебачення (у тому числі супутникові), системи (призначені для передачі по радіоканалах результатів вимірів) радіотелеметрій і тому подібне

Радіотехнічні ж системи вимірювального типа - системи радіолокацій і радіонавігаційних, системи траєкторних вимірів, системи виміру параметрів довкілля і тому подібне - відрізняються від РТС ПІ тим, що корисна інформація накладається на сигнал (або виникає в сигналі) в процесі його взаємодії з довкіллям і об'єктами і відображає параметри і властивості цих об'єктів і середовища поширення сигналів. У приймальному пункті корисна інформація витягується з сигналу шляхом виміру відповідних параметрів електромагнітного поля.

У других, визначимо: що таке інформація і в чому полягає суть передачі інформації ?

Існує досить багато визначень цього поняття, проте найбільш адекватним постановці завдання до останнього часу вважалося шенноновское визначення *інформації як заходи невизначеності* (міри незнання того, що підлягає передачі). Відповідно, мета передачі інформації - це *зняття* даної *невизначеності.*

Це класичне визначення інформації вперше було введене К. Шенноном в 1948 р. в його роботі " Математична теорія зв'язку " і на декілька десятиліть визначило підходи до аналізу і розробки методів і систем передачі інформації, які і на сьогодні значною мірою орієнтуються на це визначення. Відповідно до даного підходу у міру здобуття інформації *знімається невизначеність,* при цьому *ніж більше* інформації отримано, *тим менше* міра невизначеності одержувача.

Розвиток теорії і практики систем передачі інформації з часом виявив ряд невідповідностей між даним визначенням і отримуваними результатами, які в деяких випадках виявляються значно кращими, ніж передбачає існуюча теорія. Так, шенноновский підхід абсолютно не враховує міри корисності і свідомості *інформації*, наявність апріорних *знань про* предмет і так далі, він направлений не *на збільшення знання, а на зменшення незнання,* тобто в якомусь сенсі є пасивним. У зв'язку з цим в останні два десятиліття відбувається поступове переосмислення визначень, що здавалися непорушними, і теоретичних основ передачі інформації. В ході вивчення даного курсу будуть освітлений як класичний підхід, так і сучасний погляд на дане питання.

1.1. Модель системи передачі інформації

Вивчення систем передачі інформації почнемо з розгляду загальноприйнятих моделей, що дозволяють, абстрагуючись від приватних питань технічної реалізації конкретної системи, з'ясувати загальні принципи і закономірності їх побудови.

У найзагальнішому вигляді модель РТС ПІ можна представити таким чином (рис.1.1):

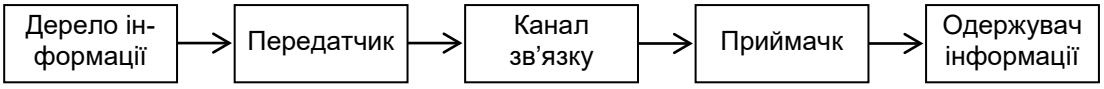


Рис.1.1

Хоча ця модель і містить основні елементи, властиві будь-якій системі передачі інформації, вона може служити лише простою ілюстрацією до опису РТС ПІ, оскільки практично не відображає тих дій, які повинні виконуватися (або можуть виконуватися) над інформацією в процесі її передачі від джерела до споживача.

Значно повнішою в цьому сенсі є модель системи передачі (і зберігання) інформації, подібна приведеною на рис.1.2, якій ми і користуватимемося надалі. Потрібно відзначити, що насправді проблеми, що виникають при передачі (причому не лише з використанням радіохвиль) і зберіганні інформації (на оптичних дисках, магнітних носіях і в пам'яті комп'ютерів) дуже схожі, тому методи їх рішення і структура технічних пристроїв також багато в чому ідентичні.

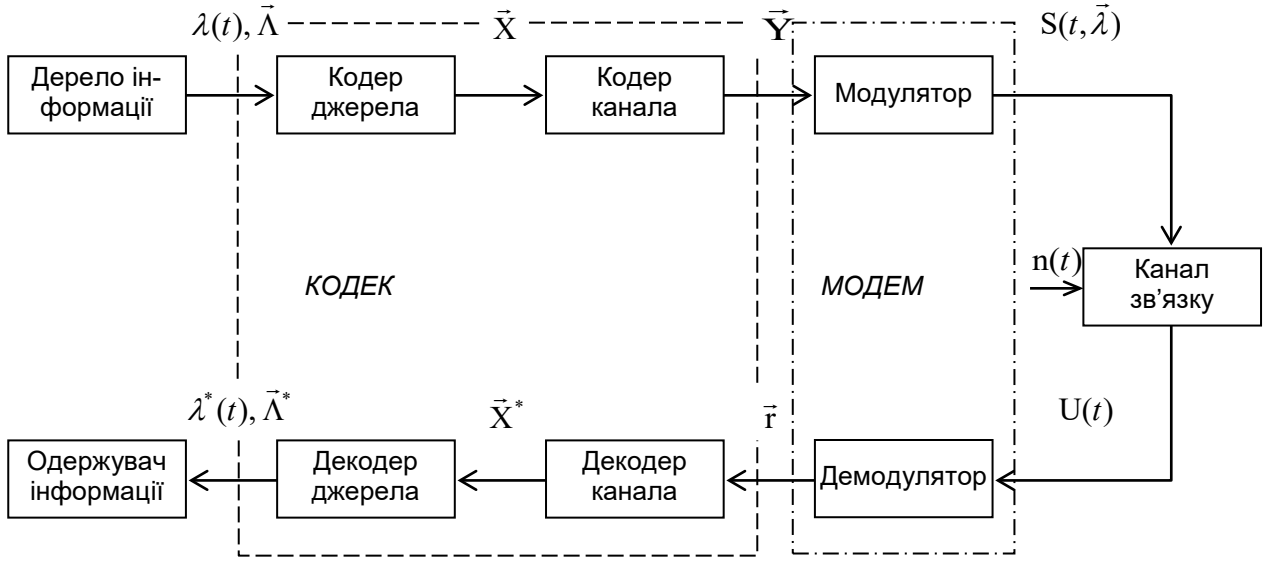


Рис.1.2

Коротко охарактеризуємо призначення і функції елементів цієї моделі.

1. *Джерело інформації або повідомлення - це фізичний об'єкт, система або явище, що формують передаване повідомлення. Само повідомлення - це значення або зміна деякої фізичної величини, що відображають стан об'єкту (системи або явища). Як правило, первинні повідомлення - мова, музика, зображення, виміри параметрів довкілля і так далі - є функції часу неелектричної природи. З метою передачі по каналу зв'язку ці повідомлення перетворяться в електричний сигнал, зміни якого в часі* $\boldsymbol{\lambda(t)}$ відображує передаване повідомлення. Значна частина передаваних повідомлень, особливо останнім часом, за своєю природою не є сигналами - це масиви чисел, текстові або інші файли і тому подібне. Повідомлення такого типа можна представити у вигляді деяких векторів $\boldsymbol{\Lambda}$.

2. *Кодер джерела. Переважна частина вихідних повідомлень - мова, музика, зображення і так далі - призначена для безпосереднього сприйняття органами чуття людини і в загальному випадку погано пристосована для їх ефективної передачі по каналах зв'язку. Тому повідомлення* ($\boldsymbol{\lambda(t)}$ або $\boldsymbol{\Lambda}$), як правило, піддаються кодуванню. У процедуру кодування зазвичай включають і дискретизацію безперервного повідомлення $\boldsymbol{\lambda(t)}$, тобто його перетворення в послідовність елементарних дискретних повідомлень $\boldsymbol{\{ \lambda_i \}}$.

Під кодуванням в загальному випадку розуміють перетворення алфавіту повідомлення $\boldsymbol{A\{ \lambda_i \}}$, ($\boldsymbol{i = 1,2…K}$) у алфавіт деяким чином вибраних кодових символів $\boldsymbol{\Re\{ x_j \}}$, ($\boldsymbol{j = 1,2…N}$). Зазвичай (але не обов'язково) розмір алфавіту кодових символів $\boldsymbol{dim\ \Re\{ x_j \}}$ менше або набагато менше розміру алфавіту джерела $\boldsymbol{dim A\{ \lambda_i \}}$. Кодування повідомлень може переслідувати різні цілі - скорочення об'єму передаваних даних (стискування даних), збільшення кількості

передаваної за одиницю часу інформації, підвищення достовірності передачі, забезпечення секретності при передачі і так далі

Під кодуванням джерела в РТС ПІ розумітимемо скорочення об'єму (стискування) інформації з метою підвищення швидкості її передачі або скорочення смуги частот, потрібних для передачі.

Кодування джерела інколи називають економним, безнадмірністю або ефективним кодуванням, а також стискуванням даних. Під ефективністю в даному випадку розуміється міра скорочення об'єму даних, що забезпечується кодуванням.

Якщо стискування виробляється так, що за стислими даними можна абсолютно точно відновити вихідну інформацію, кодування називається неруйнівним. Неруйнівне кодування використовується при передачі (або зберіганні) текстової інформації, числових даних, комп'ютерних файлів і тому подібне, тобто там, де недопустимі навіть щонайменші відмінності вихідних і відновлених даних.

У багатьох випадках немає необхідності в абсолютно точній передачі інформації від джерела до її споживача, тим більше що в каналі зв'язку завжди присутні перешкоди і абсолютно точна передача в принципі неможлива. У таких випадках може бути використане руйнівне стискування, що забезпечує відновлення вихідного повідомлення по стислому з тією або іншою мірою наближення. Як правило, руйнівні методи стискування набагато ефективніші, ніж неруйнівні.

Таким чином, на виході кодера джерела по передаваному повідомленню ***λ(t)*** або ***Λ формується послідовність кодових символів X, звана інформаційною послідовністю, допускаюча абсолютно точне (або наближене) відновлення вихідного повідомлення і що має, по можливості, як можна менший розмір.***

3. *Кодер каналу. При передачі інформації по каналу зв'язку з перешкодами в прийнятих даних можуть виникати помилки. Якщо такі помилки мають невелику величину або виникають досить рідкий, інформація може бути використана споживачем. При великому числі помилок отриманою інформацією користуватися не можна.*

Кодування в каналі, або перешкодостійке кодування, є спосіб обробки передаваних даних, що забезпечує зменшення кількості помилок, що виникають в процесі передачі по каналу з перешкодами. Існує велике число різних методів перешкодостійкого кодування інформації, але всі вони засновані на наступному: при перешкодостійкому кодуванні до передаваних повідомлень вноситься спеціальним чином організована надмірність (у передавані кодові послідовності додаються надлишкові символи), що дозволяє на приймальній стороні виявляти і виправляти виникаючі помилки. Таким чином, якщо при кодуванні джерела виробляється усунення природної надмірності, що має місце в повідомленні, то при кодуванні в каналі надмірність до передаваного повідомлення свідомо вноситься. На виході кодера каналу в результаті формується послідовність кодових символів Y (X), звана кодовою послідовністю.

Потрібно відзначити, що як перешкодостійке кодування, так і стискування даних не є обов'язковими операціями при передачі інформації. Ці процедури (і відповідні ним блоки в структурній схемі РТС ПІ) можуть бути відсутніми. Проте це може привести до дуже істотних втрат в перешкодостійкості системи, значному зменшенню швидкості передачі і зниженню якості передачі інформації. Тому практично всі сучасні системи (за виключенням, мабуть, найпростіших) повинні включати і обов'язково включають і ефективне і перешкодостійке кодування даних.

4. *Модулятор. Функції модулятора в РТС ПІ - узгодження повідомлення джерела або кодових послідовностей, що виробляються кодером, з властивостями каналу зв'язку і забезпечення можливості одночасної передачі великого числа повідомлень по загальному каналу зв'язку (яким є радіоканал).*

Дійсно, більшість безперервних $\lambda(t)$ і дискретних Λ повідомлень, що підлягають передачі, а також результати їх кодування - послідовності кодових символів X і Y - є порівняно низькочастотними сигналами з відносно широкою смугою ($\Delta F \leq 1$ МГц, $\Delta F \sim f_0$). В той же час ефективна передача з використанням електромагнітних коливань (радіохвиль) можлива лише для досить високочастотних сигналів ($f_0 \geq 1...1000$ МГц і вище) з відносно вузькосмуговими спектрами ($\Delta F << f_0$).

Тому модулятор повинен перетворити повідомлення джерела $\lambda(t)$ (Λ) або відповідні їм кодові послідовності X і Y в сигнали $S(t, \lambda(t))$, $S(t, Y(\lambda(t)))$ (накласти повідомлення на сигнали), властивості яких забезпечували б ним можливість ефективної передачі по радіоканалу (або іншим існуючим каналам зв'язку - телефонним, оптичним і так далі). При цьому сигнали, що належать безлічі систем передачі інформації, що працюють в загальному радіоканалі, мають бути такими, аби забезпечувалася незалежна передача повідомлень від всіх джерел до всіх одержувачів інформації.

На сьогодні існує велика кількість методів модуляції сигналів, що володіють різною ефективністю, забезпечують передачу інформації з тією або іншою якістю. Найпростішими з них є амплітудна, частотна і фазова модуляції безперервних сигналів. При вивченні курсу ми ознайомимося і з безліччю інших, сучасніших і значно ефективніших методів модуляції сигналів, вживаних в РТС ПІ, у тому числі що використовують широкосмугові шумоподобные сигнали. При цьому процедура модуляції розглядатиметься не просто як зміна параметрів сигналу $S(t)$ відповідно до значення передаваного повідомлення $\lambda(t)$, а як перетворення повідомлення в сигнал.

5. *Канал зв'язку. Згідно визначенню, РТС ПІ - це система передачі інформації, що використовує як її переносника від джерела до споживача електромагнітні хвилі або радіохвилі, а як середовище поширення - довколишній простір або радіоканал. У цьому, власне, і полягає головна відмінність РТС ПІ від інших систем передачі інформації, що використовують дротяні, волоконно-оптичні, акустичні і тому подібне канали. У останньому, за винятком неістотних деталей, структура таких систем і функції основних елементів ідентичні.*

Фізичні властивості радіоканалу як середовища поширення електромагнітних хвиль є предметом детального вивчення в курсі "Електродинаміка і поширення радіохвиль", ми ж розглядатимемо радіоканал у вигляді ланки РТС ПІ, на вхід якого поступає сигнал передавача ***S(t,λ(t))***, а на виході виходить сигнал ***U(t),*** який зазвичай називають прийнятим ваганням.

Існує безліч моделей радіоканалу більшої або меншої складності, проте в загальному випадку сигнал ***S(t, Y(λ(t)))***, проходя по каналу зв'язку, піддається ослабінню, набуває деякої тимчасової затримки (або фазове зрушення) і зашумляется. Вагання, що приймається ***U(t)*** в цьому випадку матиме вигляд

$$U(t) = \varepsilon S(t-\tau, Y(\lambda(t))) + n(t), \qquad (1.1)$$

де ε - затухання, τ - часове запізнювання, ***n(t)*** - шуми в каналі зв'язку.

При вивченні курсу розглянемо також особливості проходження сигналів по складніших каналах - каналах з багатопроменевим поширенням, каналам з тимчасовими і частотними завмираннями і так далі, а також особливості прийому сигналів на їх виході.

Приймач. Призначення приймача РТС ПІ - з максимально можливою точністю по прийнятому ваганню **U(t)** відтворити на своєму виході передане повідомлення **λ(t)** або Λ. Прийняте (відтворене) повідомлення із-за наявності перешкод в загальному випадку відрізняється від посланого. Прийняте повідомлення називатимемо оцінкою (мається на увазі оцінкою повідомлення) і позначати тим же символом, що і послане повідомлення, але із знаком *: $\lambda^*(t)$ або Λ^*. Процес відтворення оцінки повідомлення по прийнятому ваганню в загальному випадку включає декілька етапів.

6. *Демодулятор. Для відтворення оцінки повідомлення* $\lambda^*(t)$ або Λ^* приймач системи в першу чергу повинен по прийнятому ваганню **U(t)** і з врахуванням відомостей про використаних при передачі вигляді сигналу і способі модуляції отримати оцінку кодової послідовності ***Y*(λ(t)),*** звану прийнятою послідовністю r. Ця процедура називається демодуляцією, детектуванням або прийомом сигналу. При цьому демодуляція повинна виконуватися так, щоб прийнята послідовність r в мінімальній мірі відрізнялася від переданої кодової послідовності Y . У своїй постановці і по способах рішення завдання демодуляції прийнятого вагання U(t) в основному збігається з різними варіантами завдання оптимального прийому сигналу на тлі перешкод (оптимальне виявлення, оптимальне розрізнення два або декількох сигналів і так далі). Питання оптимального прийому сигналів в радіотехнічних системах є предметом вивчення курсу "Основи теорії РТС", який є теоретичною основою і для нашого курсу.

7. *Декодер каналу. Прийняті послідовності r в загальному випадку можуть відрізнятися від переданих кодових слів Y, тобто містити помилки. Кількість таких помилок залежить від рівня перешкод в каналі зв'яз-*

ку, швидкості передачі, вибраного для передачі сигналу і способу модуляції, а також від способу прийому (демодуляції) вагання U(t). Завдання декодера каналу - виявити і, по можливості, виправити ці помилки. Процедура виявлення і виправлення помилок в прийнятій послідовності r називається декодуванням каналу. Результатом декодування r є оцінка інформаційної послідовності $\boldsymbol{X}^*$. Вибір перешкодостійкої коди, способу кодування, а також методу декодування повинен вироблятися так, щоб на виході декодера каналу залишилося якомога менше невиправлених помилок.

Питанням перешкодостійкого кодирования/декодирования в системах передачі (і зберігання) інформації в даний час приділяється виняткова увага, оскільки цей прийом дозволяє істотно підвищити якість її передачі. У багатьох випадках, коли вимоги до достовірності інформації, що приймається, дуже великі (у комп'ютерних мережах передачі даних, в дистанційних системах управління і тому подібне), передача без перешкодостійкого кодування взагалі неможлива. В ході вивчення курсу приділимо цьому питанню особливу увагу.

8. *Декодер джерела.Оскільки інформація джерела* ($\boldsymbol{\lambda(t)}$, $\boldsymbol{\Lambda}$) в процесі передачі піддавалася кодуванню з метою її компактнішої (або зручнішого) вистави (стискування даних, економне кодування, кодування джерела), необхідно відновити її до початкового (або майже вихідному вигляду) по прийнятій послідовності $\boldsymbol{X}^*$. Процедура відновлення $\boldsymbol{\Lambda}^*$ по $\boldsymbol{X}^*$ називається декодуванням джерела і може бути або просто зворотна операції кодування (неруйнівне кодирование/декодирование), або відновлювати наближене значення $\boldsymbol{\Lambda}^*$, що більшою чи меншою мірою відрізняється від $\boldsymbol{\Lambda}$ (руйнівне кодирование/декодирование). До операції відновлення $\boldsymbol{\Lambda}^*$ по $\boldsymbol{X}^*$ відноситимемо також відновлення, якщо в цьому є необхідність, безперервній функції $\boldsymbol{\lambda}^*\mathbf{(t)}$ по набору дискретних значень оцінок $\boldsymbol{\Lambda}^*$.

Потрібно сказати, що останнім часом економне кодування займає усе більш помітне місце в системах передачі інформації, оскільки, разом з перешкодостійким кодуванням, це виявилося найефективнішим способом збільшення швидкості і якості її передачі.

Таким чином, коротко розкривши загальну структуру радіотехнічної системи передачі інформації, перейдемо до детальнішого вивчення її основних елементів. Першим з них є джерело інформації.

1.2. Джерело інформації

Джерело інформації або повідомлення - це фізичний об'єкт, система або явище, що формують передаване повідомлення. Само повідомлення - це значення або зміна деякої фізичної величини, що відображають стан об'єкту (системи або явища). Як правило, первинні повідомлення - мова, музика, зображення, виміри параметрів довкілля і так далі - є функції часу - $\boldsymbol{\lambda(t)}$ або інших аргументів - $\boldsymbol{\lambda(x, y, z)}$ неелектричної природи (акустичний тиск, температура, розподіл яскравості на деякій площині і тому подібне). З метою передачі по ка-

налу зв'язку ці повідомлення зазвичай перетворяться в електричний сигнал, зміни якого в часі ***λ(t)*** відображують передавану інформацію. Такі повідомлення називаються безперервними, або аналоговими, повідомленнями (сигналами), і для них виконуються умови

$$\lambda \in (\lambda min, \lambda max),\ t \in (0, t), \qquad \textbf{(1.2)}$$

тобто як само значення функції, так і значення аргументу для таких повідомлень безперервні або визначені для будь-якого значення безперервного інтервалу як по ***λ***, так і по ***t*** (рис. 1.3,а,б).

Багато повідомлень - команди виконавчим пристроям, телеграфні повідомлення, текстова інформація і тому подібне - носять дискретний характер. При цьому або алфавіт повідомлення $A(\lambda_i)$ є кінцевою рахунковою безліччю

$$\lambda_i = \lambda_1, \lambda_2, \ldots, \lambda_k,\ i = 1, K \qquad \textbf{(1.3)}$$

(повідомлення, дискретні або квантовані по рівню, рис.1.3,в), або самі сигнали передаються лише в дискретні моменти часу

$$t = t_1, t_2, \ldots, t_m,\ i = 1, M \qquad \textbf{(1.4)}$$

(*дискретні за часом повідомлення, рис.1.3,г), або і те і інше (дискретні за часом і по рівню сигнали, або, як їх інакше називають, цифрові сигнали, або повідомлення, рис.1.3,д, е).*

Значна частина передаваних повідомлень, особливо останнім часом, за своєю природою не є сигналами - це пакети даних, результати цифрових вимірів різних параметрів, цифрові фотографії, текстові, графічні або інші файли і тому подібне. Повідомлення такого типа можна представити у вигляді масивів чисел або деяких векторів ***Λ.***

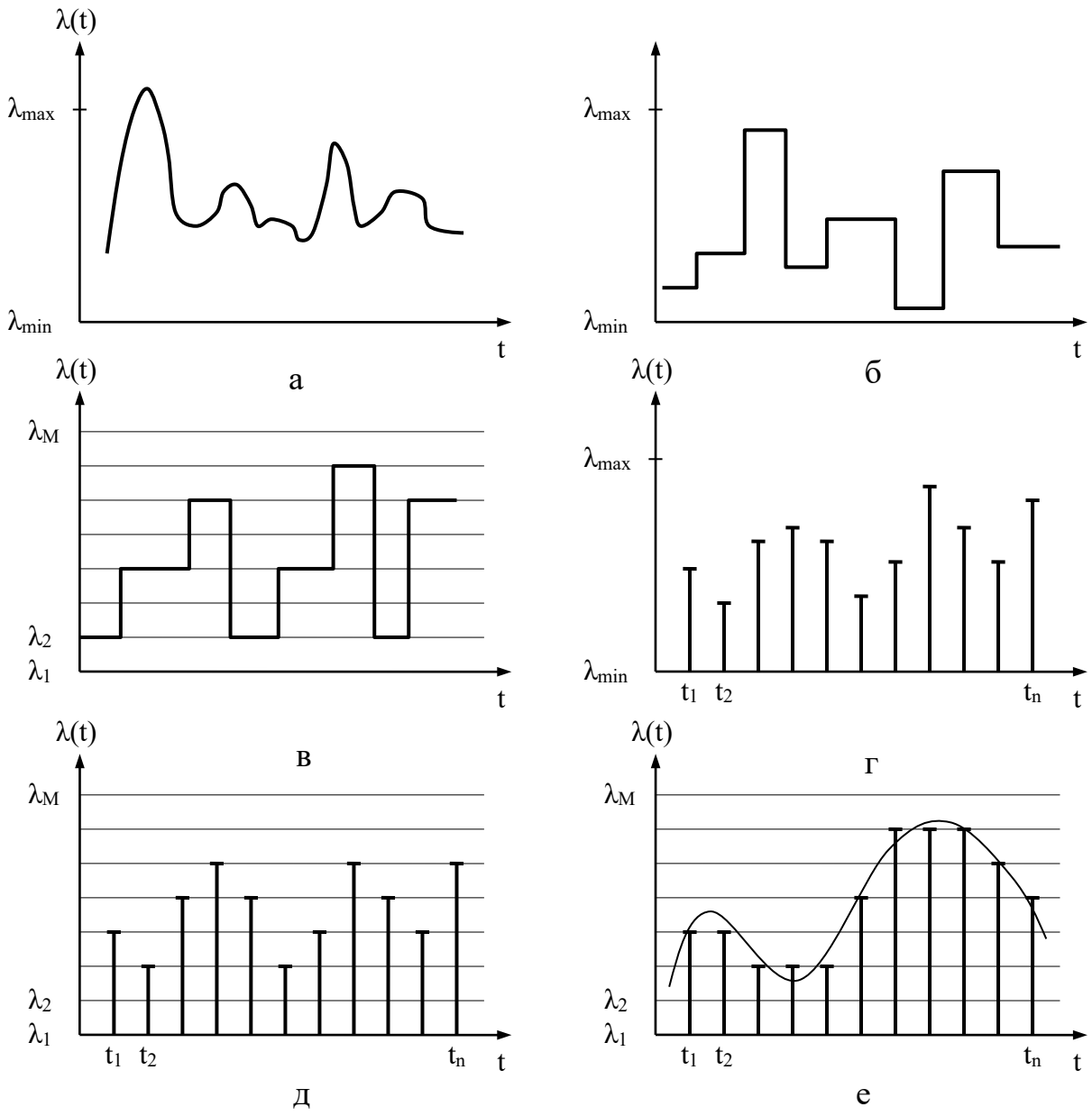


Рис. 1.3

Як випливає з наведених вище прикладів, при всій різноманітності форм що підлягають передачі повідомлень (або що відображують їх тимчасових сигналів) переважна більшість з них може бути віднесена всього лише до декількох видів, що істотно розрізняються, а саме:

- безперервні за часом (аналогові) повідомлення (сигнали);
- дискретні за часом (що дискретизують) повідомлення;
- дискретні по рівню (квантовані) повідомлення.

Виявляється, проте, що навіть такі на перший погляд зовсім різні сигнали, як безперервні і такі, що дискретизують (див. рис.1.3), мають дуже багато загального і зв'язані жорсткою функціональною залежністю, встановлюваною теоремою дискретизації, або теоремою Котельникова.

1.3. Теорема дискретизації

Виключно важливим положенням теорії зв'язку, на якому заснована вся сучасна радіотехніка, є так звана теорема відліків, або теорема Котельникова. Ця теорема дозволяє встановити співвідношення між безперервними сигналами, якими є більшість реальних інформаційних сигналів – мова, музика, електричні сигнали, відповідні телевізійним зображенням, сигнали в ланцюгах різних радіотехнічних систем і тому подібне, і значеннями цих сигналів лише в окремі моменти часу – так званими відліками. На використанні цього зв'язку будується вся сучасна цифрова радіотехніка – цифрові методи передачі і зберігання звукових і телевізійних сигналів, цифрові системи телефонного і стільникового зв'язку, системи цифрового супутникового телебачення і так далі Можна сказати більше: майбутнє всієї техніки обробки сигналів - в її цифровій реалізації. Пройдет ще 10 – 20 років - і ми згадуватимемо про традиційні аналогові методи формування і прийому сигналів, їх обробки і зберігання лише в теоретичному плані. Вся практична радіотехніка, пов'язана з обробкою інформаційних сигналів, перейде на цифрову реалізацію.

Теорема дискретизації, або, як її ще називають, теорема Котельникова, теорема Уїтекера, формулюється таким чином: безперервна функція ***X(t)*** *с обмеженим спектром, тобто що не має в своєму спектрі*

$$F\{X(t)\} = \int_{-\infty}^{\infty} X(t) \cdot e^{-j2\pi ft} dt \qquad (1.5)$$

складових з частотами, лежачими за межами смуги $f \in (-F_m, F_m)$, *повністю визначається послідовністю своїх відліків в дискретні моменти часу* $X(t_i)$, *наступних з кроком* $\Delta t < 1/F_m$.

Доведення сформульованої теореми грунтується на однозначній відповідності між сигналами і відповідними ним спектрами. Іншими словами, якщо сигнали однакові, то і відповідні ним спектри також однакові. І, навпаки, якщо спектри двох сигналів однакові, то і відповідні сигнали також однакові.

Це говорить про те, що не існує принципових відмінностей між безперервними і дискретними сигналами. Будь-який безперервний сигнал з обмеженим спектром (а всі реальні сигнали мають обмежений спектр) може бути перетворений в дискретну послідовність, а потім з абсолютною точністю відновлений по послідовності своїх дискретних значень. Останнє дозволяє також розглядати джерела безперервних повідомлень як джерела дискретних послідовностей, переходити, де це необхідно і зручно, до аналізу дискретних повідомлень, здійснювати передачу безперервних повідомлень в дискретній формі і так далі.

1.4. Кількість інформації, ентропія джерела повідомлень

Для порівняння між собою різних джерел повідомлень необхідно ввести деяку кількісну міру, яка дала б можливість об'єктивно оцінити інформацію, що міститься в повідомленні. Така міра вперше була введена К. Шеноном в 1948 р., а потім строгіше визначена А.Я. Хінчиним. Розглянемо основи інформаційного підходу Шенона.

Всяка інформація виходить споживачем після прийому повідомлення, тобто в результаті досвіду. Повідомлення, що отримується на приймальній стороні, несе корисну інформацію лише в тому випадку, якщо є невизначеність відносно стану джерела. Якщо досвід може закінчитися лише одним результатом і спостерігач заздалегідь знає результат досвіду, то по його результату він не отримує жодної інформації. Наприклад, якщо повідомлять, що сонце сходить на сході, то жодної інформації це повідомлення не принесе, оскільки всі знають, що це вірно. У такій події, як щоденний схід сонця на сході, немає нічого невизначеного, вірогідність цієї події дорівнює одиниці і кількість інформації, приношувана повідомленням про таку подію, дорівнює нулю. Інформація з'явиться лише тоді, коли джерело матиме принаймні більш за один можливий стан.

Розглянемо джерело, що видає послідовність незалежних дискретних повідомлень $\{\lambda_i\}$, кожне з яких випадковим чином вибирають з алфавіту повідомлення $A(\lambda_i) = \lambda_1, \lambda_2, \lambda_3, \ldots \lambda_K$, де K - розмір алфавіту джерела. Таке джерело називатимемо джерелом без пам'яті з кінцевим дискретним алфавітом. Повідомлення, що виробляються таким джерелом, називаються простими повідомленнями.

У кожному елементарному повідомленні λ_i для його одержувача міститься деяка інформація. Визначимо кількісну міру цієї інформації і з'ясуємо, від чого вона залежить.

До того, як зв'язок відбувся, у одержувача завжди є велика або менша невизначеність відносно того, яке повідомлення λ_i з числа можливих буде передано.

Абсолютно вочевидь, що міра цієї невизначеності, або несподіванки передачі λ_i, залежить від вірогідності передачі того або іншого повідомлення. Наприклад, якщо вірогідність передачі якого-небудь повідомлення λ_i дуже висока, то ще до передачі ми майже напевно знаємо, яке повідомлення буде передано, і його прийом не принесе нам майже жодної нової інформації.

Таким чином, вочевидь, що кількість інформації, що міститься в елементарному повідомленні λ_i, є деякою функцією від вірогідності передачі цього повідомлення $P(\lambda_i)$:

$$J(\lambda_i) = \varphi\{P(\lambda_i)\}. \qquad (1.31)$$

Визначимо вигляд цієї функції φ. Для цього зажадаємо, аби міра кількості інформації $J(\lambda_i)$ задовольняла двом інтуїтивним властивостям:

1. Якщо вибір повідомлення λ_i заздалегідь зумовлений ($P(\lambda_i) = 1$ - невизначеності немає), та кількість інформації в цьому повідомленні дорівнює нулю: $J(\lambda_i) = \varphi\{1\} = 0$.

2. Якщо джерело послідовно вибирає повідомлення λ_i і λ_j і вірогідність такого вибору $P(\lambda_i, \lambda_j)$ є спільна вірогідність подій λ_i і λ_j, то *кількість інформації*

в цих двох елементарних повідомленнях дорівнюватиме сумі кількостей інформації в кожному з них.

Вірогідність спільного випадання подій λ_i и λ_j $P(\lambda_i, \lambda_j)$, як відомо, визначається по формулі повної вірогідності

$$P(\lambda_i, \lambda_j) = P(\lambda_i) \cdot P(\lambda_j / \lambda_i) = P \cdot Q. \qquad (1.32)$$

Тоді, відповідно до вимоги (2), повинна виконуватися умова

$$\varphi \{ P \cdot Q \} = \varphi(P) + \varphi(Q). \qquad (1.33)$$

Неважко здогадатися, що функцією, що задовольняє цим двом умовам, що пред'являються до неї, є функція вигляду

$$J(\lambda_i) = a \log P(\lambda_i), \qquad (1.34)$$

при цьому як коефіцієнт а, так і підстава логарифма можуть бути вибрані довільно. Проте для зручності (аби кількісна міра інформації була позитивною) приймають а = - 1. Підставу логарифма зазвичай вибирають рівним двом, і тоді

$$J(\lambda_i) = - \log_2 P(\lambda_i). \qquad (1.35)$$

Визначена таким чином одиниця виміру інформації називається двійковою одиницею, або бітом інформації. Наприклад, якщо яке-небудь з елементарних повідомлень λ_i ***може бути вибране з алфавіту і передане з вірогідністю*** $P(\lambda_i) = 1/8$, ***то говорять, що в нім міститься*** $\log_2 (1/8) = 3$ **біта інформації.**

Інколи як підставу логарифма вибирають е, тоді інформація вимірюється в натуральних одиницях, або натах.

Кількість інформації, що міститься в одному елементарному повідомленні λ_i, ще ніяк не характеризує джерело. Одні елементарні повідомлення можуть нести багато інформації, але передаватися дуже рідко, інші - передаватися частіше, але нести менше інформації. Тому джерело може бути охарактеризоване середньою кількістю інформації, що доводиться на одне елементарне повідомлення, носить назву “Ентропія джерела” і визначуваним таким чином:

$$H(\lambda) = -\sum_{i=1}^{K} P(\lambda_i) \cdot \log P(\lambda_i), \quad i = 1, K. \qquad (1.36)$$

Ентропія, як кількісна міра інформативності джерела, володіє наступними властивостями:

1. Ентропія є величина речова, обмежена і ненегативна. Ці її властивості витікають з вигляду вираження для $H(\lambda)$, а також з урахуванням того, що $0 < P(\lambda_i) < 1$.

2. Ентропія детермінованих повідомлень дорівнює нулю, тобто $H(\lambda) = 0$, якщо хоч би одне з повідомлень має вірогідність, рівну одиниці.

3. Ентропія максимальна, якщо повідомлення λ_i рівноімовірні, тобто

$P(\lambda_1) = P(\lambda_2) = \ldots\ldots P(\lambda_k) = 1/K$, і тоді

$$H(\lambda) = -\frac{1}{K}\sum_{i=1}^{K} log\frac{1}{K} = log\, K. \qquad (1.37)$$

Як видно з останнього вираження, в разі рівноімовірних повідомлень ентропія зростає із збільшенням об'єму алфавіту джерела (зростанням числа повідомлень). При нерівноімовірних елементарних повідомленнях λ_i ентропія, відповідно, зменшується.

4. Ентропія двійкового джерела (K = 2) може змінюватися від нуля до одиниці. Дійсно, ентропія системи з двох повідомлень λ_1 и λ_2

$$\begin{aligned} H(\lambda) &= -P(\lambda_1) * log\, P(\lambda_1) - P(\lambda_2) * log\, P(\lambda_2) = \\ &= -P(\lambda_1) * log\, P(\lambda_1) - \{1 - P(\lambda_1)\} * log\{1 - P(\lambda_1)\}. \end{aligned} \qquad (1.38)$$

З останнього вираження видно, що ентропія дорівнює нулю при $P(\lambda_1) = 0$; $P(\lambda_2)=1$, або $P(\lambda_1) = 1$; $P(\lambda_2) = 0$; при цьому максимум ентропії матиме місце, коли $P(\lambda_1)=P(\lambda_2)=1/2$ і її максимальне значення дорівнюватиме 1 біт.

1.6.1. Ентропія складних повідомлень, надмірність джерела

Розглянуті вище характеристики джерела - кількість інформації і ентропія - відносилися до одного джерела, що виробляє потік незалежних або простих повідомлень, або до джерела без пам'яті.

Проте в реальних умовах незалежність елементарних повідомлень, що виробляються джерелом, - явище досить рідке. Частіше буває якраз зворотне - сильний детермінований або статистичний зв'язок між елементами повідомлення одного або декількох джерел.

Наприклад, при передачі тексту вірогідності появи окремих букв залежать від того, які букви їм передували. Для російського тексту, наприклад, якщо передана буква "П", вірогідність того, що наступною буде "А", набагато вище, ніж "Н", після букви "" ніколи не зустрічається "Н" і так далі Подібна ж картина спостерігається при передачі зображень - сусідні елементи зображення мають зазвичай майже однакову яскравість і колір.

При передачі і зберіганні даних часто також мають справу з декількома джерелами, що формують статистично зв'язані один з одним повідомлення. Повідомлення, що виробляються такими джерелами, називаються складними повідомленнями, а самі джерела - джерелами з пам'яттю.

Вочевидь, що при визначенні ентропії і кількості інформації в повідомленнях, елементи яких статистично зв'язані, не можна обмежуватися лише безумовною вірогідністю - необхідно обов'язково враховувати також умовну вірогідність появи окремих повідомлень.

Визначимо ентропію складного повідомлення, що виробляється двома залежними джерелами (подібним же чином визначається ентропія складного повідомлення, що виробляється одним джерелом з пам'яттю).

Хай повідомлення першого джерела набувають значень $x_1, x_2, x_3,....x_k$ з вірогідністю, відповідно $P(x_1), P(x_2),..... P(x_k),$ повідомлення другого - $y_1, y_2,.....y_m$ з вірогідністю $P(y_1), P(y_2),..... P(y_m).$

Спільну ентропію двох джерел X і Y можна визначити таким чином:

$$H(X,Y) = -\sum_{i=1}^{K}\sum_{j=1}^{m} P(x_i, y_j) * \log P(x_i, y_j), \qquad (1.39)$$

де $P(x_i,y_j)$ - вірогідність спільної появи повідомлень x_i і y_j. Оскільки спільна вірогідність $P(x_i,y_j)$ по формулі Байеса визначається як

$$P(x_i, y_j) = P(x_i) * P(y_j / x_i) = P(y_j) * P(x_i / y_j), \qquad (1.40)$$

те вираження для спільної ентропії можна записати в наступному вигляді:

$$H(X,Y) = -\sum_{i=1}^{k}\sum_{j=1}^{m} P(x_i) * P(y_j / x_i) * \log\{ P(x_i) * P(y_j / x_i)\} =$$

$$= -\sum_{i=1}^{k} P(x_i) * \log P(x_i) * \sum_{j=1}^{m} P(y_j / x_i) - \sum_{i=1}^{k} P(x_i) * \sum_{j=1}^{m} P(y_j / x_i) * \log P(y_j / x_i). \qquad (1.41)$$

Оскільки передачі повідомлення x_i обов'язково відповідає передача одного з повідомлень (будь-якого) з ансамблю Y, то

$$\sum_{j=1}^{m} P(y_j / x_i) = 1 \qquad (1.42)$$

і спільна ентропія H(X,Y) визначиться як

$$H(X,Y) = -\sum_{i=1}^{k} P(x_i) * \log P(x_i) - \sum_{i=1}^{k} P(x_i) * \sum_{j=1}^{m} P(y_j / x_i) * \log P(y_j / x_i) =$$

$$= H(X) + \sum_{i=1}^{k} P(x_i) * H(Y / x_i), \qquad (1.43)$$

де $H(Y/x_i)$ - так звана приватна умовна ентропія, що відображає ентропію повідомлення Y за умови, що мало місце повідомлення x_i. Другий доданок в останньому вираженні є усереднюванням $H(Y/x_i)$ по всіх повідомленнях x_i і називається середньою умовною ентропією джерела Y за умови передачі повідомлення X. І остаточно:

$$H(X,Y) = H(X) + H(Y/X). \qquad (1.44)$$

Таким чином, спільна ентропія двох повідомлень дорівнює сумі безумовної ентропії одного з них і умовної ентропії другого.

Можна відзначити наступні основні властивості ентропії складних повідомлень:

1. При статистично незалежних повідомленнях X і Y спільна ентропія дорівнює сумі ентропій кожного з джерел:

$$H(X,Y) = H(X) + H(Y) \qquad (1.45)$$

оскільки H (Y/X) = H (Y).

2. При повній статистичній залежності повідомлень X і Y спільна ентропія дорівнює безумовній ентропії одного з повідомлень. Друге повідомлення при цьому інформації не додає. Дійсно, при повній статистичній залежності повідомлень умовна вірогідність $P(y_j/x_i)$ и $P(x_i/y\ j)$ рівні або нулю, або 1, тоді

$$P(x_i/y_j)*\log P(x_i/y_j) = P(y_j/x_i)*\log P(y_j/x_i) = 0 \qquad (1.46)$$

і, отже, $H(X,Y) = H(X) = H(Y)$.

3. Умовна ентропія змінюється в межах

$$0 < H(Y/X) < H(Y). \qquad (1.47)$$

4. Для спільної ентропії двох джерел завжди справедливе співвідношення

$$H(X,Y) \le H(X) + H(Y), \qquad (1.48)$$

при цьому умова рівності виконується лише для незалежних джерел повідомлень.

Отже, за наявності зв'язку між елементарними повідомленнями ентропія джерела знижується, причому в тим більшій мірі, чим сильніше зв'язок між елементами повідомлення.

Таким чином, можна зробити наступні виводи відносно міри інформативності джерел повідомлень:

1. *Ентропія джерела і кількість інформації тим більше, чим більше розмір алфавіту джерела.*

2. Ентропія джерела залежить від статистичних властивостей повідомлень. Ентропія максимальна, якщо повідомлення джерела рівноімовірні і статистично незалежні.

3. *Ентропія джерела, що виробляє нерівноімовірні повідомлення, завжди менше максимально досяжною.*

4. *За наявності статистичних зв'язків між елементарними повідомленнями (пам'яті джерела) його ентропія зменшується.*

Як приклад розглянемо джерело з алфавітом, що складається з букв російської мови ***а ,б, в,.....,ю, я***. Вважатимемо для простоти, що розмір алфавіту джерела $K = 2^5 = 32$.

Якби всі букви російського алфавіту мали однакову вірогідність і були статистично незалежні, то середня ентропія, що доводиться на один символ, склала б

$$H(\lambda)_{max} = \log_2 32 = 5 \text{ біт/букву.}$$

Якщо тепер врахувати лише різну вірогідність букв в тексті (а неважко перевірити, що так воно і є), розрахункова ентропія складе

$$H(\lambda) = 4{,}39 \text{ біт/букву.}$$

З врахуванням кореляції (статистичному зв'язку) між двома і трьома сусідніми буквами (після букви "П" частіше зустрічається "А" і майже ніколи – "Ю" і "Ц") ентропія зменшиться, відповідно, до

$$H(\lambda) = 3{,}52 \text{ біт/букву} \text{ и } H(\lambda) = 3{,}05 \text{ біт/букву}.$$

Нарешті, якщо врахувати кореляцію між вісьма і більш символами, ентропія зменшиться до

$$H(\lambda) = 2{,}0 \text{ біт/букву}$$

і далі залишається без змін.

У зв'язку з тим, що реальні джерела з одним і тим же розміром алфавіту можуть мати абсолютно різну ентропію (а це не лише тексти, але і мова, музика, зображення і так далі), вводять таку характеристику джерела, як надмірність

$$\rho_u = 1 - H(\lambda) / H(\lambda)_{max} = 1 - H(\lambda)/\log K, \qquad (1.49)$$

де $H(\lambda)$ - ентропія реального джерела, $\log K$ - максимально досяжна ентропія для джерела з об'ємом алфавіту в До символів.

Тоді, наприклад, надмірність літературного російського тексту складе

$$\rho_u = 1 - (2 \text{ біта/букву})/(5 \text{ біт/букву}) = 0{,}6 .$$

Іншими словами, при передачі тексту по каналу зв'язку кожні шість букв з десяти передаваних не несуть жодної інформації і можуть без всяких втрат просто не передаватися.

Такий же, якщо не вищою (ρ_u= 0,9...0,95) надмірністю володіють і інші джерела інформації - мова, і особливо музика, телевізійні зображення і так далі

Виникає законне питання: чи потрібно займати носій інформації або канал зв'язку передачею символів, що практично не несуть інформації, або ж можливе таке перетворення вихідного повідомлення, при якому інформація "втискувалася" б в мінімально необхідне для цього число символів?

Виявляється, не лише можна, але і необхідно. Сьогодні багато хто з існуючих радіотехнічних систем передачі інформації і зв'язку просто не зміг би працювати, якби в них не вироблялося такого роду кодування. Не було б циф-

рового стільникового зв'язку стандартів GSM і CDMA. Не працювали б системи цифрового супутникового телебачення, дуже неефективної була б робота Internet, а вже про те, аби поглянути відеофільм або послухати хорошу музику з лазерного диска, не могло бути і мови. Все це забезпечується ефективним або економним кодуванням інформації в даних системах.

Вивченню цього розділу сучасної радіотехніки – основ теорії і техніки економного, або безнадмірності, кодування - і присвячена наступна частина нашого курсу.

2. Основи економного кодування

Повідомлення, передавані з використанням РТС ПІ (мова, музика, телевізійні зображення і так далі) в більшості своїй призначені для безпосереднього сприйняття органами чуття людини і зазвичай погано пристосовані для їх ефективної передачі по каналах зв'язку. Тому вони в процесі передачі, як правило, піддаються кодуванню.

Що таке кодування і навіщо воно використовується?

Під кодуванням в загальному випадку розуміють перетворення алфавіту повідомлення $A\{\lambda_i\}$, ($i = 1,2\ldots K$) у алфавіт деяким чином вибраних кодових символів $\mathcal{R}\{x_j\}$, ($j = 1,2\ldots N$). Зазвичай (але не обов'язково) розмір алфавіту кодових символів $\dim \mathcal{R}\{x_j\}$ менше або набагато менше розміру алфавіту джерела $\dim A\{\lambda_i\}$.

Кодування повідомлень може переслідувати різні цілі. Наприклад, це кодування з метою засекречування передаваної інформації. При цьому елементарним повідомленням λ_i з алфавіту $A\{\lambda_i\}$ ставляться у відповідність послідовності, наприклад, цифр або букв із спеціальних кодових таблиць, відомих лише відправникові і одержувачеві інформації.

Іншим прикладом кодування може служити перетворення дискретних повідомлень λ_i з одних систем числення в інших (з десяткової в двійкову, вісімкову і т. п., з непозиційної в позиційну, перетворення буквеного алфавіту в цифровій і т. д.).

Кодування в каналі, або перешкодостійке кодування інформації, може бути використано для зменшення кількості помилок, що виникають при передачі по каналу з перешкодами.

Нарешті, кодування повідомлень може вироблятися з метою скорочення об'єму інформації і підвищення швидкості її передачі або скорочення смуги частот, потрібних для передачі. Таке кодування називають економним, безнадмірністю, або ефективним кодуванням, а також стискуванням даних. У даному розділі йтиме мова саме про такого роду кодуванні. Процедурі кодування зазвичай передують (і включаються в неї) дискретизація і квантування безперервного повідомлення $\lambda(t)$, тобто його перетворення в послідовність елементарних дискретних повідомлень $\{\lambda_{iq}\}$.

Перш ніж перейти до питання економного кодування, коротко пояснимо суть самої процедури кодування.

Будь-яке дискретне повідомлення λ_i *з алфавіту джерела* $A\{\lambda_i\}$ об'ємом в K символів можна закодувати послідовністю відповідним чином вибраних кодових символів x_j з алфавіту $\mathcal{R}\{x_j\}$.

Наприклад, будь-яке число (а λ_i можна вважати числом) можна записати в заданной позиційній системі числення таким чином:

$$\lambda_i = M = x_{n-1} \cdot m^{n-1} + x_{n-2} \cdot m^{n-2} + \ldots + x_0 \cdot m^0, \qquad (2.1)$$

де m - підстава системи числення; $x_0 \ldots x_{n-1}$ - коефіцієнти при степенях m; $x \subset 0, m-1$.

Хай, наприклад, значення $\lambda_i = M = 59$, тоді його код по основі $m = 8$, матиме вигляд

$$M = 59 = 7 \cdot 8^1 + 3 \cdot 8^0 = 73_8 .$$

Код того ж числа, але при поснові m = 4 виглядатиме таким чином:

$$M = 59 = 3 \cdot 4^2 + 2 \cdot 4^1 + 3 \cdot 4^0 = 323_4 .$$

Нарешті, якщо основа кода $m = 2$, то

$$M = 59 = 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 111011_2 .$$

Таким чином, числа *73, 323 і 111011* можна вважати, відповідно, вісімковим, четверичным і двійковим кодами числа $M = 59$.

В принципі підстава коди може бути будь-яким, проте найбільшого поширення набули двійкові коди, або коди з основою 2, для яких розмір алфавіту кодових символів $\Re\{x_j\}$ рівний двом, $x_j \subset 0,1$. Двійкові коди, тобто коди, нулі, що містять лише, і одиниці, дуже просто формуються і передаються по каналах зв'язку і, головне, є внутрішньою мовою цифрових ЕОМ, тобто без всяких перетворень можуть оброблятися цифровими засобами. Тому, коли йдеться про кодуванні і кодах, найчастіше мають на увазі именно двійкові коди. Надалі розглядатимемо в основному двійкове кодування.

Найпростішим способом вистави або завдання код є кодові таблиці, що ставлять у відповідність повідомленням λ_i відповідні ним коди (табл. 2.1).

Таблиця 2.1

Буква λ_i	Число λ_i	Код с основою 10	Код с основою 4	Код с основою 2
А	0	0	00	000
Б	1	1	01	001
В	2	2	02	010
Г	3	3	03	011
Д	4	4	10	100
Е	5	5	11	101
Ж	6	6	12	110
З	7	7	13	111

Іншим наочним і зручним способом опису код є їх вистава у вигляді кодового дерева (рис.2.1).

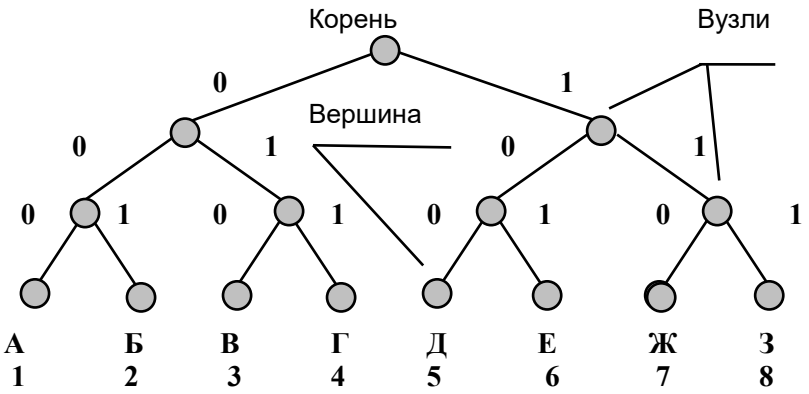


Рис. 2.1

Для того, щоб побудувати кодове дерево для даної коди, починаючи з деякої крапки - кореня кодового дерева - проводяться гілки - 0 або 1. На вершинах кодового дерева знаходяться букви алфавіту джерела, причому кожній букві відповідають своя вершина і своя дорога від кореня до вершини. Наприклад, букві А відповідає код 000, букві В – 010, букві Е – 101 і так далі

Код, отриманий з використанням кодового дерева, змальованого на рис.2.1, є рівномірним трьохрозрядним кодом.

Рівномірні коди дуже широко використовуються через свою простоту і зручність процедур кодування-декодування: кожній букві – однакове число біт; прийняв задане число біт – шукай в кодовій таблиці відповідну букву.

Разом з рівномірними кодами можуть застосовуватися і нерівномірні коди, коли кожна буква з алфавіту джерела кодується різним числом символів, наприклад, А - 10, Б – 110, В – 1110 і так далі

Кодове дерево для нерівномірного кодування може виглядати, наприклад, так, як показано на рис.2.2.

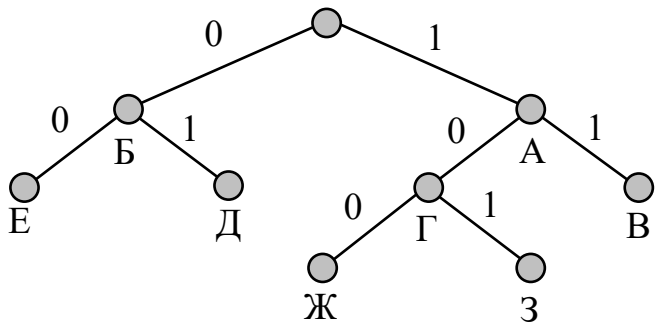


Рис. 2.2

При використанні цієї коди буква А кодуватиметься, як 1, Би - як 0, В – як 11 і так далі Проте можна відмітити, що, закодувавши, наприклад, текст АБ-БА = 1001, ми не зможемо його однозначно декодувати, оскільки такий же код дають фрази: ЖА = 1001, АЄА = 1001 і ГД = 1001. Такі коди, що не забезпечують однозначного декодування, називаються такими, що наводяться, або

непрефіксними, кодами і не можуть на практиці застосовуватися без спеціальних розділяючих символів. Прикладом вживання такого типа код може служити азбука Морзе, в якій окрім крапок і тире є спеціальні символи розділення букв і слів. Але це вже не двійковий код.

Проте можна побудувати нерівномірні коди, що не наводяться, допускають однозначне декодування. Для цього необхідно, аби всім буквам алфавіту відповідали лише вершини кодового дерева, наприклад, такого, як показано на рис.2.3. Тут жодна кодова комбінація не є початком інший, довшою, тому неоднозначності декодування не буде. Такі нерівномірні коди називаються префіксними.

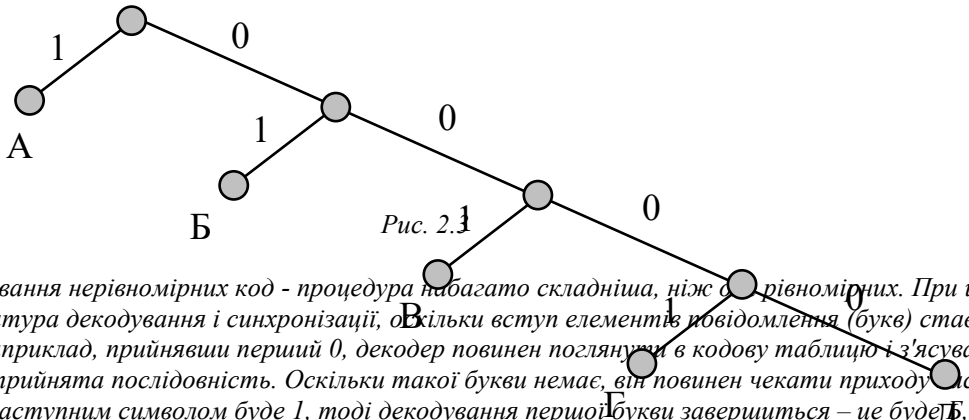


Рис. 2.3

Прийом і декодування нерівномірних код - процедура набагато складніша, ніж для рівномірних. При цьому ускладнюється апаратура декодування і синхронізації, оскільки вступ елементів повідомлення (букв) стає нерегулярним. Так, наприклад, прийнявши перший 0, декодер повинен поглянути в кодову таблицю і з'ясувати, якій букві відповідає прийнята послідовність. Оскільки такої букви немає, він повинен чекати приходу наступного символу. Якщо наступним символом буде 1, тоді декодування першої букви завершиться – це буде Б, якщо ж другим прийнятим символом знову буде 0, доведеться чекати третього символу і так далі

Навіщо ж використовуються нерівномірні коди, якщо вони настільки незручні?

Розглянемо приклад кодування повідомлень λ_i з алфавіту об'ємом K = 8 за допомогою довільного n-разрядного двійкового коду.

Хай джерело повідомлення видає деякий текст з алфавітом від А до З і однакової вірогідності букв $\mathbf{P(\lambda_i) = 1/8.}$

Кодуючий пристрій кодує ці букви рівномірним трьохрозрядним кодом (див. таблиці. 2.1).

Визначимо основні інформаційні характеристики джерела з таким алфавітом:

- ентропія джерела $H(\bar{\lambda}) = -\sum_{i=1}^{K} P_i \log P_i$, $H(\bar{\lambda}) = \log K = 3$;
- надмірність джерела $\rho_и = 1 - \frac{H(\bar{\lambda})}{\log K} = 0$;
- середнє число символів в коді $\bar{n} = \sum_{i=1}^{K} n_i \cdot P_i = \sum_{i=1}^{8} 3 \cdot \frac{1}{8} = 3$;
- надмірність коди $\rho_k = 1 - \frac{H(\bar{\lambda})}{\bar{n}} = 0$.

Таким чином, при кодуванні повідомлень з рівноімовірними буквами надмірність вибраної (рівномірного) коди виявилася рівною нулю.

Хай тепер вірогідність появи в тексті різних букв буде різною (таблиця. 2.2).

Таблиця 2.2

А	Б	В	Г	Д	Е	Ж	З
$P_а$=0.6	$P_б$=0.2	$P_в$=0.1	$P_г$=0.04	$P_д$=0.025	$P_е$=0.015	$P_ж$=0.01	$P_з$=0.01

Ентропія джерела $H(\bar{\lambda}) = -\sum_{i=1}^{K} P_i \log P_i$ в цьому випадку, природно, буде меншою і складе $H(\bar{\lambda})$ = 1.781.

Середнє число символів на одне повідомлення при використанні тієї ж рівномірного трьохрозрядного коду

$$\bar{n} = \sum_{i=1}^{K} n_i P_i = n \sum_{i=1}^{K} P_i = n = 3.$$

Надмірність коди в цьому випадку буде

$$\rho_k = 1 - \frac{H(\bar{\lambda})}{\bar{n}} = 1 - \frac{1.781}{3} \approx 0.41,$$

або досить значною величиною (в середньому 4 символи з 10 не несуть жодної інформації).

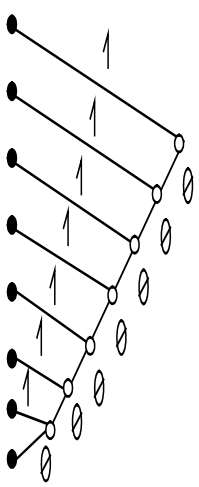
У зв'язку з тим, що при кодуванні нерівноімовірних повідомлень рівномірні коди володіють великою надмірністю, було запропоновано використовувати нерівномірні коди, тривалість кодових комбінацій яких була б погоджена з вірогідністю випадання різних букв.

Таке кодування називається статистичним.

Нерівномірний код при статистичному кодуванні вибирають так, щоб вірогідніші букви передавалися за допомогою коротших комбінацій коди, менш вірогідні - за допомогою довших. В результаті зменшується середня довжина кодової групи порівняно з випадком рівномірного кодування.

Один із способів такого кодування запропонований Хаффменом. Побудова кодового дерева нерівномірної коди Хаффмена для передачі одного з вось-ми повідомлень λ_i з різною вірогідністю ілюструється таблиця. 2.3.

Таблиця 2.3

Буква	Вірогідність P_i	Кодове дерево	Код	n_i	$n_i \cdot P_i$
А	0.6		1	1	0.6
Б	0.2		01	2	0.4
В	0.1		001	3	0.3
Г	0.04		0001	4	0.16
Д	0.025		00001	5	0.125
Е	0.015		000001	6	0.08
Ж	0.01		0000001	7	0.07
З	0.01		00000001	8	0.08

Середнє число символів для такого коду складе

$$\overline{n} = \sum_{i=1}^{8} n_i P_i = 1.825,$$

а надмірність коду

$$\rho_k = 1 - \frac{H(\overline{\lambda})}{\overline{n}} = 1 - \frac{1.781}{1.825} \approx 0.03,$$

тобто на порядок менше, чим при рівномірному кодуванні.

Іншим простим способом статистичного кодування є кодування по методу Шеннона-Фано. Кодування відповідно до цього алгоритму виробляється так:

- спочатку всі букви з алфавіту повідомлення записують в порядку убування їх вірогідності;
- потім всю сукупність букв розбивають на дві приблизно рівні по сумі вірогідності групи; однією з них (у групі може бути будь-яке число символів, у тому числі – один) привласнюють символ “1”, інший - “0”;
- кожну з цих груп знову розбивають (якщо це можливо) на дві частини і кожною з частин привласнюють “1” і “0” і так далі

Процедура кодування по методу Шеннона-Фано ілюструється таблиці. 2.4.

Таблиця 2.4

Буква	$P(\lambda_i)$	I	II	III	IV	V	Код	$n_i \cdot P_i$
А	0.6	1					1	0.6
Б	0.2		1	1			011	0.6
В	0.1			0			010	0.3
Г	0.04			1			001	0.12
Д	0.025	0			1		0001	0.1
Е	0.015		0	0			00001	0.075
Ж	0.01				0	1	000001	0.06
З	0.01					0	000000	0.06

Для отриманої таким чином коди середнє число двійкових символів, що доводяться на одну букву, рівне

$$\overline{n} = \sum_{i=0}^{7} n_i P_i \approx 1.9,$$

а надмірність коди складе

$$\rho_k = 1 - \frac{1.781}{1.9} \approx 0.06,$$

тобто також істотно меншу величину, ніж для рівномірної коди.

Звернемо увагу на той факт, що як для коди Хаффмена, так і для коди Шеннона-Фано середня кількість двійкових символів, що доводиться на символ джерела, наближається до ентропії джерела, але не дорівнює їй. Даний результат є наслідком теореми кодування без шуму для джерела (першої теореми Шенона), яка стверджує:

Будь-яке джерело можна закодувати двійковою послідовністю при середній кількості двійкових символів на символ джерела λ_i*, скільки завгодно близькому до ентропії, і неможливо добитися середньої довжини коди, меншої, ніж ентропія* $H(\lambda)$.

Значення цієї теореми для сучасної радіотехніки важко переоцінити – вона встановлює ті кордони в компактності представлення інформації, яких можна досягти при правильному її кодуванні.

2.1. Мета стискування даних і типи систем стискування

Передача і зберігання інформації вимагають досить великих витрат. І чим з великою кількістю інформації нам доводиться мати справу, тим дорожче це стоїть. На жаль, велика частина даних, які потрібно передавати по каналах зв'язку і зберігати, має не найкомпактнішу виставу. Швидше, ці дані зберігаються у формі, що забезпечує їх найбільш просте використання, наприклад: звичайні книжкові тексти, ASCII коди текстових редакторів, двійкові коди даних ЕОМ, окремі відліки сигналів в системах збору даних і так далі Проте таке найбільш просте у використанні представлення даних вимагає удвічі - втричі, а інколи і в сотні разів більше місця для їх збереження і смуга частот для їх передачі, ніж насправді потрібно. Тому стискування даних – це один з найбільш актуальних напрямів сучасної радіотехніки.

Таким чином, мета стискування даних - забезпечити компактне представлення даних, що виробляються джерелом, для їх економнішого збереження і передачі по каналах зв'язку.

Враховуючи надзвичайну важливість процедури економного кодування даних при їх передачі, виділимо її з узагальненої схеми РТС ПІ і детально розглянемо в справжньому розділі нашого курсу.

Нижче приведена умовна структура системи стискування даних:

Дані джерела(Кодер(Стислі дані(Декодер(Відновлені дані

У цій схемі що виробляються джерелом дані визначимо як дані джерела, а їх компактна вистава - як стислі дані. Система стискування даних складається з кодера і декодера джерела. Кодер перетворить дані джерела в стислі дані, а декодер призначений для відновлення даних джерела із стислих даних. Відновлені дані, що виробляються декодером, можуть або абсолютно точно збігатися з вихідними даними джерела, або трохи відрізнятися від них.

Існують два типи систем стискування даних:

- системи стискування без втрат інформації (неруйнівне стискування);
- системи стискування з втратами інформації (руйнівне стискування).

2.1.1. Стискування без втрат інформації

У системах стискування без втрат декодер відновлює дані джерела абсолютно точно, таким чином, структура системи стискування виглядає таким чином:

Вектор даних **X** → Кодер → **B (X)** → Декодер → **X**

Вектор даних джерела X, що підлягають стискуванню, є послідовність $\boldsymbol{X} = (\boldsymbol{x_1, x_2,\ldots x_n})$ кінцевої довжини. Відліки $\boldsymbol{x_i}$ - складові вектора X - вибрані з кінцевого алфавіту даних A. При цьому розмір вектора даних n обмежений, але він може бути скільки завгодно великим. Таким чином, джерело на своєму виході формує як дані X послідовність довжиною n з алфавіту A .

Вихід кодера - стислі дані, відповідні вхідному вектору X, - представимо у вигляді двійкової послідовності $\boldsymbol{B(X) = (b_1,b_2,\ldots b_k),}$ розмір якої $\boldsymbol{k}$ залежить від **X**. Назвемо **B(X)** кодовим словом, привласненим вектору **X** кодером (або кодовим словом, в яке вектор X перетворений кодером). Оскільки система стискування - неруйнівна, однаковим векторам $\boldsymbol{X_l = X_m}$ должны соответствовать одинаковые кодовые слова $\boldsymbol{B(X_l) = = B(X_m).}$

При рішенні задачі стискування природним є питання, наскільки ефективна та або інша система стискування. Оскільки, як ми вже відзначали, в основному використовується лише двійкове кодування, то такою мірою може служити коефіцієнт стискування r, визначуваний як відношення

$$r = \frac{\text{розмір даних джерела в бітах}}{\text{розмір стислих даних в бітах}} = \frac{n \log_2 (\dim A)}{k}, \qquad (2.2)$$

де $\boldsymbol{dim\ A}$ **-** розмір алфавіту даних ***A.***

Таким чином, коефіцієнт стискування r = 2 означає, що об'єм стислих даних складає половину від об'єму даних джерела. Чим більше коефіцієнт стискування r, тим краще працює система стискування даних.

Разом з коефіцієнтом стискування r ефективність системи стискування може бути охарактеризована швидкістю стискування R, визначуваною як відношення

$$\boldsymbol{R = k/n} \qquad \boldsymbol{(2.3)}$$

і вимірюваною в кількості кодових біт, що доводяться на відлік даних джерела". Система, що має більший коефіцієнт стискування, забезпечує меншу швидкість стискування.

2.1.2. Стискування з втратою інформації

У системі стискування з втратами (або з руйнуванням) кодування виробляється таким чином, що декодер не в змозі відновити дані джерела в первинному вигляді. Структурна схема системи стискування з руйнуванням виглядає таким чином:

X → Квантователь → $\mathbf{X^q}$ → Неразрушающий кодер → **B** ($\mathbf{X^q}$) → Декодер → **X***

Як і в попередній схемі, $X = (x_1, x_2,\ldots x_n)$ - вектор даних, що підлягають стискуванню. Відновлений вектор позначимо як $X^* = (x_1, x_2,\ldots x_n)$. Відзначимо наявність в цій схемі стискування елементу, який був відсутній при неруйнівному стискуванні, - квантователя.

Квантователь стосовно вектора вхідних даних X формує вектор X^q, досить близький до X в сенсі среднеквадратического відстані. Робота квантователя заснована на пониженні розміру алфавіту (простий квантователь виробляє округлення даних до найближчого цілого числа).

Далі кодер піддає неруйнівному стискуванню вектор квантованих даних X^q таким чином, що забезпечується однозначна відповідність між X^q и $B(X^q)$ (для $X_l^q = X_m^q$ виконується умова $B(X_l^q) = B(X_m^q)$). Проте система в цілому залишається руйнівною, оскільки двом різним векторам X може відповідати один і той же вектор X^*.

Руйнівний кодер характеризується двома параметрами - швидкістю стискування R і величиною спотворень D, визначуваних як

$$R = k/n,$$

$$D = (1/n) \sum(x_i - x_i)^2. \qquad (2.4)$$

Параметр R характеризує швидкість стискування в бітах на один відлік джерела, величина D є мірою среднеквадратического відмінності між X^* і X.

Якщо є система руйнівного стискування з швидкістю і спотвореннями R_1 і D_1 відповідно і друга система з швидкістю R_2 і спотвореннями D_2, то перша з них краще, якщо $R_1 < R_2$ и $D_1 < D_2$. Проте, на жаль, неможливо побудувати систему руйнівного стискування, що забезпечує одночасне зниження швидкості R і зменшення спотворень D, оскільки ці два параметри зв'язано зворотною залежністю. Тому метою оптимізації системи стискування з втратами може бути або мінімізація швидкості при заданій величині спотворень, або здобуття найменших спотворень при заданій швидкості стискування.

Вибір системи неруйнівного або руйнівного стискування залежить від типа даних, що підлягають стискуванню. При стискуванні текстових даних, комп'ютерних програм, документів, креслень і тому подібне абсолютно вочевидь, що потрібно застосовувати неруйнівні методи, оскільки необхідне абсолютно точне відновлення вихідної інформації після її стискування. При стискуванні мови, музичних даних і зображень, навпаки, частіше використовується руйнівне стискування, оскільки при практично непомітних спотвореннях воно забезпечує на порядок, а інколи і на два меншу швидкість R. У загальному випадку руйнівне стискування забезпечує, як правило, істотно вищі коефіцієнти стискування, ніж неруйнівне.

Нижче приведені ряд прикладів, що ілюструють необхідність процедури стискування, прості методи економного кодування і ефективність стискування даних.

Прииклад 1. Передбачимо, що джерело генерує цифрове зображення (кадр) розміром 512*512 елементи, що містить 256 кольорів. Кожен колір є числом з безлічі {0,1,2. 255}. Математично це зображення є матрицею 512x512, кожен елемент якої належить безлічі {0,1,2. 255}. (Елементи зображення називають пікселами).

У свою чергу, кожен піксел з безлічі {0,1,2. 255} може бути представлений в двійковій формі з використанням 8 біт. Таким чином, розмір даних джерела в бітах складе 8x512x512= 2^{21}, или 2,1 Мегабіта.

На жорсткий диск об'ємом в 1 Гігабайт поміститься приблизно 5000 кадрів зображення, якщо вони не піддаються стискуванню (відеоролик тривалістю приблизно в п'ять хвилин). Якщо ж це зображення піддати стискуванню з коефіцієнтом r = 10, то на цьому ж диску ми зможемо зберегти вже майже годинний відеофільм!

Передбачимо далі, що ми хочемо передати вихідне зображення по телефонній лінії, пропускна спроможність якої складає 14000 бит/с. На це доведеться витратити 21000000 бит/14000 бит/с, або приблизно 3 хвилини. При стискуванні ж даних з коефіцієнтом r = 40 на це піде всього 5 секунд !

Приклад 2. Як дані джерела, що підлягають стискуванню, виберемо фрагмент зображення розміром 4x4 елементу і що містить 4 кольори: R = ="красный", O = "помаранчевий", Y = "синій", G = "зелений":

R	R	O	Y
R	O	O	Y
O	O	Y	G
Y	Y	Y	G

Проскануємо це зображення по рядках і кожному з кольорів привласнимо відповідну інтенсивність, наприклад, R = 3, O = 2, Y = 1 і G = 0, внаслідок чого отримаємо вектор даних X = (3,3,2,1,3,2,2,1,2,2,1,0,1,1,1,0).

Для стискування даних візьмемо кодер, що використовує наступну таблицю того, що перекодувало даних джерела в кодові слова (питання про вибір таблиці залишимо на майбутнє) :

Кодер	
Відлік	Кодове слово
3	001
2	01
1	1
0	000

Використовуючи таблицю кодування, замінимо кожен елемент вектора X відповідною кодовою послідовністю з таблиці (так зване кодування без пам'яті). Стислі дані (кодове слово B(X)) виглядатимуть таким чином:

$$B(X)= (0,0,1,0,0,1,0,1,1,0,0,1,0,1,0,1,1,0,1,0,1,1,0,0,0,1,1,1,0,0,0).$$

Коефіцієнт стискування при цьому складе r = 32/31, або 1,03. Відповідно швидкість стискування R = 31/16 біт на відлік.

Приклад 3. Порівняємо два різні кодери, що здійснюють стискування одного і того ж вектора даних

$$X = ABRACADABRA .$$

Перший кодер - кодер без пам'яті, аналогічний розглянутому в попередньому прикладі (кожен елемент вектора X кодується незалежно від значень інших елементів - кодер без пам'яті). Таблиця кодування для нього виглядає таким чином:

Кодер 1	
Символ	Кодове слово
A	0
B	10
R	110
C	1110
D	1111

Другий кодер при кодуванні поточного символу враховує значення передування йому символу, таким чином, кодове слово для поточного символу A буде різним в поєднаннях RA, DA і CA (іншими словами, код володіє пам'яттю в один символ джерела):

Кодер 2	
Символ, попередній символ	Кодове слово
(A,-)	1
(B,A)	0
(C,A)	10
(D,A)	11
(A,R)	1
(R,B)	1
(A,C)	1
(A,B)	1

Кодові слова, відповідні вектору даних X = ABRACADABRA, при кодуванні з використанням цих двох таблиць матимуть вигляд:

$$B_1(X) = 01011001110011110101100,$$

$$B_2(X) = 10111011111011.$$

Таким чином, швидкість стискування при використанні кодера 1 (без пам'яті) складе 23/11 = 2,09 біта на символ даних, тоді як для кодера 2 - 13/11 = =1,18 біта на символ. Використання другого кодера, отже, є переважнішим, хоча він і складніший.

Простими кодами, на основі яких може виконуватися стискування даних, є коди без пам'яті. У коді без пам'яті кожен символ в кодованому векторі даних замінюється кодовим словом з префіксної безлічі двійкових послідовностей або слів.

Префіксною множининою двійкових послідовностей S називається кінцева множина двійкових послідовностей, таких, що жодна послідовність в цій множиніі не є префіксом, або початком жодній іншій послідовності в S.

Наприклад, множина двійкових слів S1 = {00, 01, 100, 110, 1010, 1011 } є префіксним безліччю двійкових послідовностей, оскільки, якщо перевірити будь-яку з 30 можливих спільних комбінацій (w_i w_j) із ***S1,*** то видно, що w_i ніколи не з'явиться префіксом (або початком) w_j. З іншого боку, безліч S2 = { 00, 001, 1110 } не є префіксним безліччю двійкових послідовностей, оскільки послідовність 00 є префіксом (початком) іншої послідовності з цієї множини - 001.

Таким чином, якщо необхідно закодувати деякий вектор даних $X = (x_1, x_2, \dots x_n)$ з алфавітом даних A розміру до, те кодування кодом без пам'яті здійснюється таким чином:

- складають повний список символів $a_1, a_2, a_j \dots, a_k$ алфавіта A , в якому a_j - j-й по частоті появи в X символ, тобто першим в списку стоятиме найбільш символ, що часто зустрічається в алфавіті, другим – що рідше зустрічається і т.д.;
- кожному символу a_j назначають кодовое слово w_j з префіксної безлічі двійкових послідовностей S;
- вихід кодера отримують об'єднанням в одну послідовність всіх отриманих двійкових слів.

Формування префіксної безлічі і робота з ними – це окрема серйозна тема з теорії безлічі, що виходить за рамки нашого курсу, але декілька необхідних зауважень все-таки доведеться зробити.

Якщо $S = \{ w_1, w_2, \dots , w_k \}$ - префіксна безліч, то можна визначити деякий вектор $v(S) = (L_1, L_2, \dots , L_k)$, що складається з чисел, відповідних префіксних послідовностей, що є довжинами (L_i - довжина w_i).

Вектор ($L_1, L_2, \dots , L_k$), що складається з позитивних цілих чисел, що не зменшуються, називається вектором Крафта. Для нього виконується нерівність

$$2^{-L_1} + 2^{-L_2} + \dots + 2^{-L_k} \le 1 . \qquad (2.5)$$

Ця нерівність називається нерівністю Крафта. Для нього справедливо наступне твердження: якщо S - яка-небудь префіксна безліч, то v(S) - вектор Крафта.

Іншими словами, довжини двійкових послідовностей в префіксній безлічі задовольняють нерівності Крафта.

Якщо нерівність (2.5) переходить в строгу рівність, то такий код називається компактним і володіє найменшою серед код з даним алфавітом довжиною, тобто є оптимальним.

Нижче наведені приклади простіших префіксних множин і відповідні ним вектори Крафта:

S1 = {0, 10, 11} и v(S1) = (1, 2, 2);

S2 = {0, 10, 110, 111} и ***v(S2) = (1, 2, 3, 3);***

S3 = {0, 10, 110, 1110, 1111} и ***v(S3) = (1, 2, 3, 4, 4);***

S4 = {0, 10, 1100, 1101, 1110, 1111} и ***v(S4) = (1, 2, 4, 4, 4, 4);***

S5 = {0, 10, 1100, 1101, 1110, 11110, 11111} и ***v(S5) = (1, 2, 4, 4, 4, 5, 5);***

S6 = {0, 10, 1100, 1101, 1110, 11110, 111110, 111111}

и ***v(S6) = (1,2,4,4,4,5,6,6).***

Допустимо ми хочемо розробити код без пам'яті для стискування вектора даних $X = (x_1, x_2, \dots x_n)$ з алфавітом А розміром в до символів. Введемо в розгляд так званий вектор частот $F = (F_1, F_2, \dots , F_k)$, де F_i - кількість появ і-го найбільш символу, що часто зустрічається, з А в Х. Закодуємо Х кодом без пам'яті, для якого вектор Крафта $L = (L_1, L_2, \dots , L_k)$.

Тоді довжина двійкової кодової послідовності В(Х) на виході кодера складе

$$L*F = L_1*F_1 + L_2*F_2 + \dots + L_k*F_k. \qquad (2.6)$$

Кращим кодом без пам'яті був би код, для якого довжина В(Х), - мінімальна. Для розробки такої коди нам потрібно знайти вектор Крафта L, для якого твір L*F був би мінімальним.

Простий перебір можливих варіантів - взагалі-то, не самий кращий спосіб знайти такий вектор Крафта, особливо для великого до.

Алгоритм Хаффмена, названий на честь його винахідника - Девіда Хаффмена, - дає нам ефективний спосіб пошуку оптимального вектора Крафта L для F, тобто такого L, для якого точковий твір L*F – мінімально. Код, отриманий з використанням оптимального L для F, називають кодом Хаффмена.

2.2.1. Алгоритм Хаффмена

Алгоритм Хаффмена витончено реалізує загальну ідею статистичного кодування з використанням префіксної безлічі і працює таким чином:

1. Виписуємо в ряд всі символи алфавіту в порядку зростання або убування вірогідності їх появи в тексті.

2. Послідовно об'єднуємо два символи з найменшою вірогідністю появи в новий складений символ, вірогідність появи якого вважаємо рівній сумі вірогідності складових його символів. Врешті-решт побудуємо дерево, кожен вузол якого має сумарну вірогідність всіх вузлів, що знаходяться нижче за нього.

3. Прослідкуємо дорогу до кожного аркуша дерева, позначаючи напрям до кожного вузла (наприклад, направо - 1, наліво - 0) . Отримана послідовність дає кодове слово, відповідне кожному символу (рис.2.4).

Побудуємо кодове дерево для повідомлення з наступним алфавітом:

A	**B**	**C**	**D**	**E**
10	5	8	13	10

B	**C**	**A**	**E**	**D**
5	8	10	10	13

A	**E**	**BC**	**D**
10	10	13	13

BC	**D**	**AE**
13	13	20

AE	**BCD**
20	26

AEBCD

46

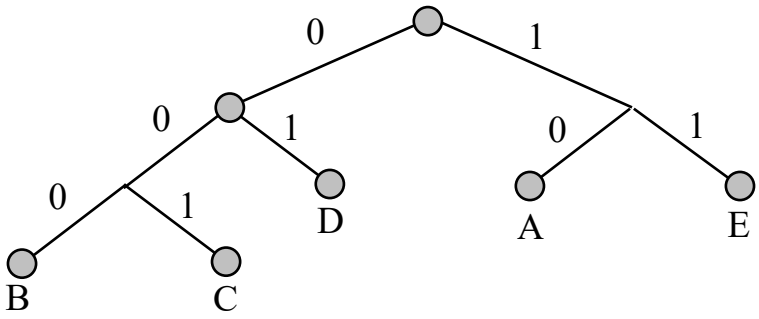


Рис. 2.4

2.2.2. Кордони ентропії для коди Хаффмена

Нагадаємо, що сформульована раніше теорема Шенона для каналу без перешкод визначає верхній кордон міри стискування для будь-яких код через ентропію вектора X . Знайдемо верхній і нижній кордони міри стискування для коди Хаффмена.

Хай R(X) - швидкість стискування, визначувана як відношення

$$\boldsymbol{R = k/n}\ , \tag{2.7}$$

вимірювана в бітах (числі двійкових символів) на букву і отримувана при кодуванні вектора даних X з використанням коди Хаффмена. Необхідно знайти кордони для R(X), виражені через ентропію вектора X.

Хай X - вектор даних довжиною n і $\boldsymbol{(F_1, F_2, \ldots , F_k)}$ - вектор частот символів в $\boldsymbol{X}$. Ентропія H(X) визначається як

$$\boldsymbol{H(X) \triangleq \frac{1}{n} \sum_{i=1}^{k} F_i \log_2 \left(\frac{n}{F_i} \right)}. \tag{2.8}$$

Спробуємо пов'язати ентропію H(X) з мірою (швидкістю) стискування R(X) таким чином:

$$R(X) = \frac{1}{n}\sum_{i=1}^{k} F_i L_i, \qquad (2.9)$$

де ($L_1, L_2, \ldots, L_k$) - точний вектор Крафта кода Хаффмена для X.

Крива $y = \log_e X$ - опукла, тобто вона завжди мається в своєму розпорядженні нижчим за будь-яку власну дотичну лінію, виключаючи точку дотику. Зокрема, вона знаходиться нижчим за пряму лінію $y = x - 1$, за винятком $x = 1$, оскільки ця пряма лінія - дотична до кривої $y = \log_e X$ в точке $x = 1$. Таким чином, можно записати нерівність

$$\log_e X \le x - 1, \qquad x > 0, \qquad x \ne 1. \qquad (2.10)$$

Підставимо $x = n \cdot 2^{-L_i} / F_i$ в (2.10) і, умножаючи обоє сторони на Fi/n, отримаємо

$$(F_i / n) \log_e \left(\frac{2^{-L_i}}{F_i / n}\right) \le 2^{-L_i} - F_i / n. \qquad (2.11)$$

Підсумуємо обоє частини нерівності по i. Сума в правій частині дорівнюватиме нулю, оскільки і сума 2^{-L_i} и сума Fi/n дають одиницю.

Отже

$$-\sum_{i=1}^{k} (F_i / n) \log_e (F_i / n) + \sum_{i=1}^{k} (F_i / n) \log_e (2^{-L_i}) \le 0. \qquad (2.12)$$

Нарешті, привівши логарифм по підставі e до підстави 2, отримаємо

$$H(X) \le R(X). \qquad (2.13)$$

Розглянемо вектор Крафта ($L1^*, L2^* \ldots, Lk^*$), в якому довжини кодових слів Li^* пов'язані з частотами символів таким чином:

$$L_i^* \triangleq [\, -\log_2 (F_i / n) \,]. \qquad (2.14)$$

Тоді

$$R(X) = n^{-1} \sum_i F_i L_i \le n^{-1} \sum_i F_i L_i^* \le n^{-1} \sum_i F_i \left[-\log_2 (F_i / n) + 1 \right], \qquad (2.15)$$

де враховано, що [u] ≤ $u + 1$. Права частина нерівності - це $H(X) + 1$.

І остаточно, об'єднуючи все вышеполученное разом

$$H(X) \le R(X) \le H(X) + 1. \qquad (2.16)$$

Недоліки методу Хаффмена

Найбільшою складністю з кодами Хаффмена, як випливає з попереднього обговорення, є необхідність мати таблиці вірогідності для кожного типа стискуваних даних. Це не представляє проблеми, якщо відомо, що стискується англійський або російський текст; ми просто надаємо кодеру і декодеру відповідне для англійського або російського тексту кодове дерево. Загалом же випадку, коли вірогідність символів для вхідних даних невідома, статичні коди Хаффмена працюють неефективно.

Вирішенням цієї проблеми є статистичний аналіз кодованих даних, що виконується в ході першого проходу за даними, і складання на його основі кодового дерева. Власне кодування при цьому виконується другим проходом.

Існує, правда, динамічна версія стискування Хаффмена, яка може будувати дерево Хаффмена "на льоту" під час читання і активного стискування. Дерево постійно оновлюється, аби відображати зміни вірогідності вхідних даних. Проте і вона на практиці володіє серйозними обмеженнями і недоліками і, крім того, забезпечує меншу ефективність стискування.

Ще один недолік код Хаффмена - це те, що мінімальна довжина кодового слова для них не може бути менше одиниці, тоді як ентропія повідомлення сповна може складати і 0,1, і 0,01 бит/букву. В цьому випадку код Хаффмена стає істотно надлишковим. Проблема вирішується застосуванням алгоритму до блоків символів, але тоді ускладнюється процедура кодирования/декодирования і значно розширюється кодове дерево, яке потрібно зрештою зберігати разом з кодом.

Нарешті, код Хаффмена забезпечує середню довжину коди, співпадаючу з ентропією, лише у тому випадку, коли вірогідність символів джерела є цілими негативними мірами двійки: 1/2 = 0,5; 1/4 = 0,25; 1/8 = 0,125; 1/16 = 0,0625 і так далі На практиці ж така ситуація зустрічається дуже рідко або може бути створена блокуванням символів зі всіма витікаючими звідси наслідками.

2.3. Коди з пам'яттю

Зазвичай розглядають двох типів код з пам'яттю:

- *блокові коди;*

- *коди з кінцевою пам'яттю.*

Блоковий код ділить вектор даних на блоки заданої довжини і потім кожен блок замінюють кодовим словом з префіксної безлічі двійкових слів. Отриману послідовність кодових слів об'єднують в результуючий двійковий рядок на виході кодера. Про блоковий код говорять, що він - блоковий код к-го порядку, якщо всі блоки мають довжину, рівну до.

Як приклад стискуватимемо вектор даних X = (0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1), використовуючи блоковий код другого порядку.

Спочатку розіб'ємо вектор X на блоки довжини 2: 01, 10, 10, 01, 11, 10, 01, 01.

Розглядатимемо ці блоки як елементи нового “гіпералфавіту” { 01, 10, 11 }. Аби визначити, який код призначити якому їх символів цього нового алфа-

віту, визначимо вектор частот появ елементів додаткового алфавіту в послідовності блоків. Отримаємо вектор частот (4, 3, 1), тобто найбільш блок, що часто зустрічається, 01 з'являється 4 рази, наступний по частоті тієї, що зустрічається блок 10 - три рази і найменш блок, що часто зустрічається, 11 - лише один раз.

Оптимальний вектор Крафта для вектора частот (4, 3, 1) - це вектор (1, 2, 2). Таким чином, кодер для оптимальної блокової коди другого порядку стосовно заданого вектора даних X визначається таблиця. 2.5.

Таблиця 2.5

Таблиця кодера	
Блок	Кодове слово
01	0
10	10
11	11

Замінюючи кожен блок привласненим йому кодовим словом з таблиці кодера, отримаємо послідовність кодових слів:

0, 10, 10, 0, 11, 10, 0, 0.

Об'єднуючи ці кодові слова разом, маємо вихідну послідовність кодера:

B(X)= (0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0).

Можна перевірити, що ентропія H(X) вихідного вектора X рівна 0.9887 бит/букву, а міра стискування, що отримується в результаті використання блокової коди R(X)= 12/16 = 0.75 біт на вибірку даних, виявилася менше нижнього кордону ентропії. Отриманий результат, здавалося б, противоречит теоремі Шенона, що стверджує, що неможливо досягти середньої довжини коди, меншої ентропії джерела. Проте насправді це не так. Якщо уважно поглянути на вектор даних X, то можна відмітити закономірність: за символом 0 частіше слідує 1, тобто умовна вірогідність P(1/0) істотно більша, ніж просто P(0). Отже, ентропію цього джерела потрібно рахувати як ентропію складного повідомлення, а вона, як відомо, завжди менше, ніж для джерела простих повідомлень.

Код з кінцевою пам'яттю при кодуванні вектора даних ***(X_1, X_2, ..., X_n)*** використовує кодову книгу, що складається з декількох різних код без пам'яті. Кожна вибірка даних кодується кодом без пам'яті з кодової книги, визначуваним значенням деякого числа попередніх вибірок даних.

Як приклад кодування кодом з пам'яттю розглянемо вже згадуваний раніше класичний приклад X = ABRACADABRA. У цій послідовності абсолютно очевидна сильна статистична залежність між її черговими символами, яка повинна обов'язково враховуватися при виборі методу кодування.

Кодер з пам'яттю при кодуванні поточного символу враховує значення передування йому символу. Таким чином, кодове слово для поточного символу A буде різним в поєднаннях RA, DA і CA (іншими словами, код володіє пам'яттю в один символ джерела) (таблиця. 2.6).

Таблиця 2.6

Кодер	
Символ, попередній символ	Кодове слово

(A,-)	1
(B,A)	0
(C,A)	10
(D,A)	11
(A,R)	1
(R,B)	1
(A,C)	1
(A,B)	1

Результат кодування - вектор B(X)= (10111011111011) завдовжки в 11 біт і швидкістю стискування R = 13/11=1,18 біта на символ даних, тоді як при кодуванні рівномірним трьохрозрядним кодом з R = 3 знадобилося б 33 біта.

2.4. Арифметичне кодування

При арифметичному кодуванні, на відміну від розглянутих нами методів, коли кодований символ (або група символів) замінюється відповідним ним кодом, результат кодування всього повідомлення представляється одним або парою дійсних чисел в інтервалі від 0 до 1. У міру кодування вихідного тексту інтервал, що відображує його, зменшується, а кількість десяткових (або двійкових) розрядів, службовців для його вистави, зростає. Чергові символи вхідного тексту скорочують величину інтервалу виходячи із значень їх вірогідності, визначуваної моделлю. Вірогідніші символи роблять це у меншій мірі, чим менш вірогідні, і, отже, додають менше розрядів до результату.

Пояснимо ідею арифметичного кодування на простому прикладі. Хай нам потрібно закодувати наступний текстовий рядок: РАДІОВІЗИР.

Перед початком роботи кодера відповідний кодованому тексту вихідний інтервал складає [0; 1).

Алфавіт кодованого повідомлення містить наступні символи (букви): { Р, А, Д, І, Про, В, З }.

Визначимо кількість (зустрічається, вірогідність) кожного з символів алфавіту в повідомленні і призначимо кожному з них інтервал, пропорційний його вірогідності. З урахуванням того, що в кодованому слові всього 10 букв, отримаємо таблицю. 2.7.

Таблиця 2.7

Символ	Вірогідність	Інтервал
А	0.1	0 – 0.1
Д	0.1	0.1 – 0.2
В	0.1	0.2 – 0.3
И	0.3	0.3 – 0.6
З	0.1	0.6 – 0.7
О	0.1	0.7 – 0.8
Р	0.2	0.8 – 1

Розташовувати символи в таблиці можна у будь-якому порядку: у міру їх появи в тексті, в алфавітному або за збільшенням вірогідності – це абсолютно не принципово. Результат кодування при цьому буде різним, але ефект – однаковим.

2.4.1. Кодування

Отже, перед початком кодування вихідний інтервал складає [0 – 1).

Після перегляду першого символу повідомлення Р кодер звужує вихідний інтервал до нового - [0.8; 1), який модель виділяє цьому символу. Таким чином, після кодування першої букви результат кодування знаходитиметься в інтервалі чисел [0.8 - 1).

Наступним символом повідомлення, що поступає в кодер, буде буква А. Еслі б ця буква була першою в кодованому повідомленні, їй був би відведений інтервал [0 - 0.1), але вона слідує за Р і тому кодується новим подынтервалом усередині вже виділеного для першої букви, звужуючи його до величини [0.80 - 0.82). Іншими словами, інтервал [0 - 0.1), виділений для букви А, розташовується тепер усередині інтервалу, займаного попереднім символом (почало і кінець нового інтервалу визначаються шляхом збільшення до початку попереднього інтервалу твору ширини попереднього інтервалу на значення інтервалу, відведені поточному символу). В результаті отримаємо новий робочий інтервал [0.80 - 0.82), оскільки попередній інтервал мав ширину в 0.2 одиниць і одна десята від нього є 0.02.

Наступному символу Д відповідає виділений інтервал [0.1 - 0.2), що стосовно вже наявного робочого інтервалу [0.80 - 0.82) звужує його до величини [0.802 - 0.804).

Наступним символом, що поступає на вхід кодера, буде буква І з виділеним для неї фіксованим інтервалом [0,3 – 0,6). Стосовно вже наявного робочого інтервалу отримаємо [0,8026 - 0,8032).

Продовжуючи в тому ж дусі, маємо:

спочатку		[0.0 - 1.0)
після перегляду	***Р***	[0.8 - 1.0)
	А	[0.80 - 0.82)
	Д	[0.802 - 0.804)
	И	[0.8026 - 0.8032)
	О	[0.80302 - 0.80308)
	В	[0.803032 - 0.803038)
	И	[0.8030338 - 0.8030356)
	З	[0.80303488 - 0.80303506)
	И	[0.803034934 - 0.803034988)
	Р	[0.8030349772 - 0.8030349880)

Результат кодування: інтервал [0,8030349772 – 0,8030349880]. Насправді, для однозначного декодування тепер досить знати лише одну кордон інтервалу – нижнюю або верхню, тобто результатом кодування може служити початок кінцевого інтервалу - 0,8030349772. Якщо бути ще точнішим, то будь-яке число, поміщене усередині цього інтервалу, однозначно декодується у вихідне повідомлення. Наприклад, це можна перевірити з числом 0,80303498, що задовольняє цим умовам. При цьому останнє число має менше число десяткових розрядів, чим числа, відповідні нижньому і верхньому кордонам інтервалу, і, отже може бути представлено меншим числом двійкових розрядів.

Неважко переконатися в тому, що, чим ширше кінцевий інтервал, тим меншим числом десяткових (і, отже, двійкових) розрядів він може бути представлений. Ширіна ж інтервалу залежить від розподілу вірогідності кодованих символів – вірогідніші символи звужують інтервал у меншій мірі і, отже, додають до результату кодування менше біт. Покажемо це на простому прикладі.

Допустимо, нам потрібно закодувати наступний рядок символів: A A A A A A A A A A #, де вірогідність букви A складає 0,9. Процедура кодування цього рядка і отримуваний результат виглядатимуть в цьому випадку таким чином:

Вхідний символ	Нижній кордон	Верхній кордон
	0.0	1.0
A	0.0	0.9
A	0.0	0.81
A	0.0	0.729
A	0.0	0.6561
A	0.0	0.59049
A	0.0	0.531441
A	0.0	0.4782969
A	0.0	0.43046721
A	0.0	0.387420489
#	0.3486784401	0.387420489

Результатом кодування тепер може бути, наприклад, число 0.35, що цілком потрапляє всередину кінцевого інтервалу 0.3486784401 – 0.387420489. Для двійкового представлення цього числа нам знадобиться 7 біт (два десяткові розряди відповідають приблизно семи двійковим), тоді як для двійкового представлення результатів кодування з попереднього прикладу – 0,80303498 – потрібне 27 біт !!!

Алгоритм кодування для послідовності довільної довжини може виглядати приблизно таким чином:

```
Set low to 0.0
Set high to 1.0
While there are still input symbols do
   get an input symbol
   code_range = high - low.
   high = low + range*high_range(symbol)
   low = low + range*low_range(symbol)
End of While
output low
```

2.4.2. Декодування

Передбачимо, що все що декодер знає про текст, – це кінцевий інтервал [0,8030349772 - 0,8030349880]. Декодеру, як і кодеру, відома також таблиця розподілу виділених алфавіту інтервалів. Він відразу ж розуміє, що перший закодований символ є P, оскільки результат кодування цілком лежить в інтервалі [0.8 - 1), виділеному моделлю символу P згідно таблиці .

Тепер повторимо дії кодера:

спочатку [0.0 - 1.0);

після перегляду [0.8 - 1.0).

Виключимо з результату кодування вплив тепер уже відомого першого символу P, для цього віднімемо з результату кодування нижній кордон діапазону, відведеного для P, – 0,8030349772 – 0.8 = =0,0030349772 – і розділимо от-

риманий результат на ширину інтервалу, відведеного для Р, – 0.2. В результаті отримаємо 0,0030349772 / 0,2 = =0,015174886. Це число цілком вміщається в інтервал, відведений для букви А,, – [0 – 0,1), отже, другим символом декодованої послідовності буде А.

Оскільки тепер ми знаємо вже дві декодовані букви - РА, виключимо з підсумкового інтервалу вплив букви А. Для цього віднімемо із залишку 0,015174886 нижній кордон для букви А 0,015174886 – 0.0 = =0,015174886 і розділимо результат на ширину інтервалу, відведеного для букви А, тобто на 0,1. В результаті отримаємо 0,015174886/0,1=0,15174886. Результат лежить в діапазоні, відведеному для букви Д, отже, чергова буква буде Д.

Виключимо з результату кодування вплив букви Д. Получим (0,15174886 – 0,1) /0,1 = 0,5174886. Результат потрапляє в інтервал, відведений для букви І, отже, черговий декодований символ – І, і так далі, поки не декодуємо всі символи:

Декодоване число	Символ на виході	Кордону нижня	Кордону верхня	Ширіна інтервалу
0,8030349772	***Р***	0.8	1.0	0.2
0,015174886	***А***	0.0	0.1	0.1
0,15174886	***Д***	0.1	0.2	0.1
0,5174886	***И***	0.3	0.6	0.3
0,724962	***О***	0,7	0,8	0,1
0,24962	***В***	0,2	0,2	0,1
0,4962	***И***	0,3	0,6	0,3
0,654	***З***	0,6	0,7	0,1
0,54	***И***	0,3	0,6	0,3
0,8	***Р***	0,6	0,8	0,2
0.0	Кінець декодування			

Описаний алгоритм декодування виглядає таким чином:

```
get encoded number
Do
   find symbol whose range straddles the encoded number
   output the symbol
   range = symbol low value - symbol high value
   subtract symbol low value from encoded number
   divide encoded number by range
until no more symbols
```

Це основна ідея арифметичного кодування, його практична реалізація дещо складніше. Деякі проблеми можна відмітити безпосередньо з наведеного прикладу.

Перша полягає в тому, що декодеру потрібно обов'язково яким-небудь чином дати знати про закінчення процедури декодування, оскільки залишок 0,0 може означати букву а або послідовність аа, ааа, аааа і так далі Вирішити цю проблему можна двома способами.

По-перше, окрім коди даних можна зберігати число, що є розміром кодованого масиву. Процес декодування в цьому випадку буде припинений, як тільки масив на виході декодера стане такого розміру.

Інший спосіб – включити в модель джерела спеціальний символ кінця блоку, наприклад #, і припиняти декодування, коли цей символ з'явиться на виході декодера.

Друга проблема витікає з самої ідеї арифметичного кодування і полягає в тому, що остаточний результат кодування – кінцевий інтервал – стане відомий лише після закінчення процесу кодування. Отже, не можна почати передачу закодованого повідомлення, поки не отримана остання буква вихідного повідомлення і не визначений остаточний інтервал? Насправді в цій затримці немає необхідності. У міру того, як інтервал, що представляє результат кодування, звужується, старші десяткові знаки його (або старші біти, якщо число записується в двійковій формі) перестають змінюватися (поглянете на наведений приклад кодування). Отже, ці розряди (або біти) вже можуть передаватися. Таким чином, передача закодованої послідовності здійснюється, хоча і з деякою затримкою, але остання незначна і не залежить від розміру кодованого повідомлення.

І третя проблема – це питання точності вистави. З наведеного прикладу видно, що точність представлення інтервалу (число десяткових розрядів, потрібне для його вистави) необмежено зростає при збільшенні довжини кодованого повідомлення. Ця проблема зазвичай вирішується використанням арифметики з кінцевою розрядністю і відстежуванням переповнювання розрядності регістрів.

2.5. Словарні методи кодування. Метод Зіва-Лемпела

Практично всі словарні методи кодування належать сім'ї алгоритмів з роботи двох ізраїльських учених - Зіва і Лемпела, опублікованою в 1977 році. Єство їх полягає в тому, що фрази в стискуваному тексті замінюються покажчиком на те місце, де вони в цьому тексті вже раніше з'являлися.

Це сімейство алгоритмів називається методом Зіва-Лемпела і позначається як LZ-сжатие. Цей метод швидко пристосовується до структури тексту і може кодувати короткі функціональні слова, оскільки вони дуже часто в нім з'являються. Нові слова і фрази можуть також формуватися з частин раніше зустрінутих слів.

Декодування стислого тексту здійснюється безпосередньо - відбувається проста заміна покажчика готовою фразою із словника, на яку той вказує. На практиці LZ-метод добивається хорошого стискування, його важливою властивістю є дуже швидка робота декодера. (Коли ми говоримо про текст, то передбачаємо, що кодуванню піддається деякий вектор даних з кінцевим дискретним алфавітом, і це не обов'язково текст в буквальному розумінні цього слова.)

Більшість словарних методів кодування носять ім'я авторів ідеї методу Зіва і Лемпела, і часто вважають, що всі вони використовують один і той же алгоритм кодування. Насправді різні представники цього сімейства алгоритмів дуже сильно розрізняються в деталях своєї роботи.

Всі словарні методи кодування можна розбити на дві групи.

Методи, що належать до першої групи, знаходячи в кодованій послідовності ланцюжки символів, які раніше вже зустрічалися, замість того, щоб повторювати ці ланцюжки, замінюють їх покажчиками на попередні повторення.

Словник в цій групі алгоритмів в неявному вигляді міститься в оброблюваних даних, зберігаються лише покажчики на ланцюжки символів, що повторюються, що зустрічаються.

Всі методи цієї групи базуються на алгоритмі, розробленому і опублікованому, як вже наголошувалося, порівняно недавно - в 1977 році Абрамом Лемпелем і Якобом Зівом, - LZ77. Найбільш досконалим представником цієї групи, що включив всі досягнення, отримані в даному напрямі, є алгоритм LZSS, опублікований в 1982 році Сторером і Шиманськи.

Процедура кодування відповідно до алгоритмів цієї групи ілюструється рис.2.5.

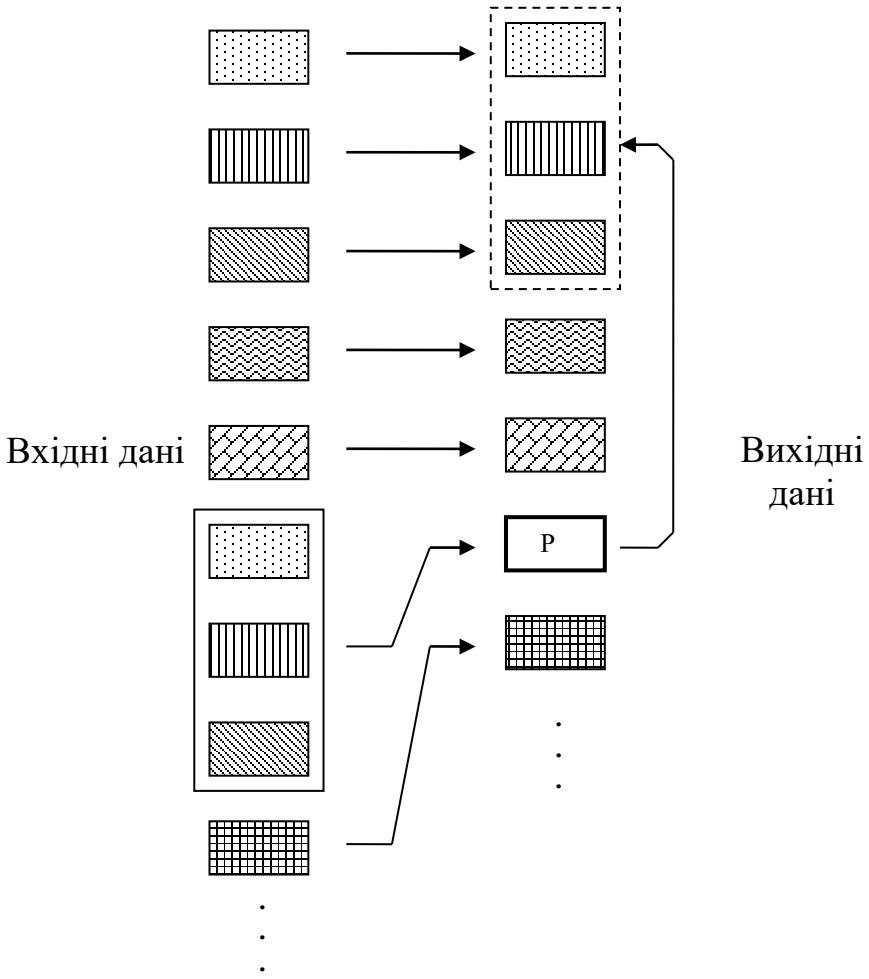


Рис. 2.5

Алгоритми другої групи на додаток до вихідного словника джерела створюють словник фраз, що є комбінаціями символів вихідного словника, що повторюються, зустрічаються у вхідних даних. При цьому розмір словника джерела зростає, і для його кодування буде потрібно більше число біт, але значна частина цього словника буде вже не окремими буквами, а букводрукувальними або цілі слова. Коли кодер виявляє фразу, яка раніше вже зустрічалася, він замінює її індексом словника, що містить цю фразу. При цьому довжина коди індексу виходить менше або набагато менше довжини коди фрази.

Всі методи цієї групи базуються на алгоритмі, розробленому і опублікованому Лемпелем і Зівом в 1978 році, – LZ78. Найбільш досконалим на даний момент представником цієї групи словарних методів є алгоритм LZW, розроблений в 1984 році Тері Велчем.

Ідею цієї групи алгоритмів можна також пояснити за допомогою рис.2.6.

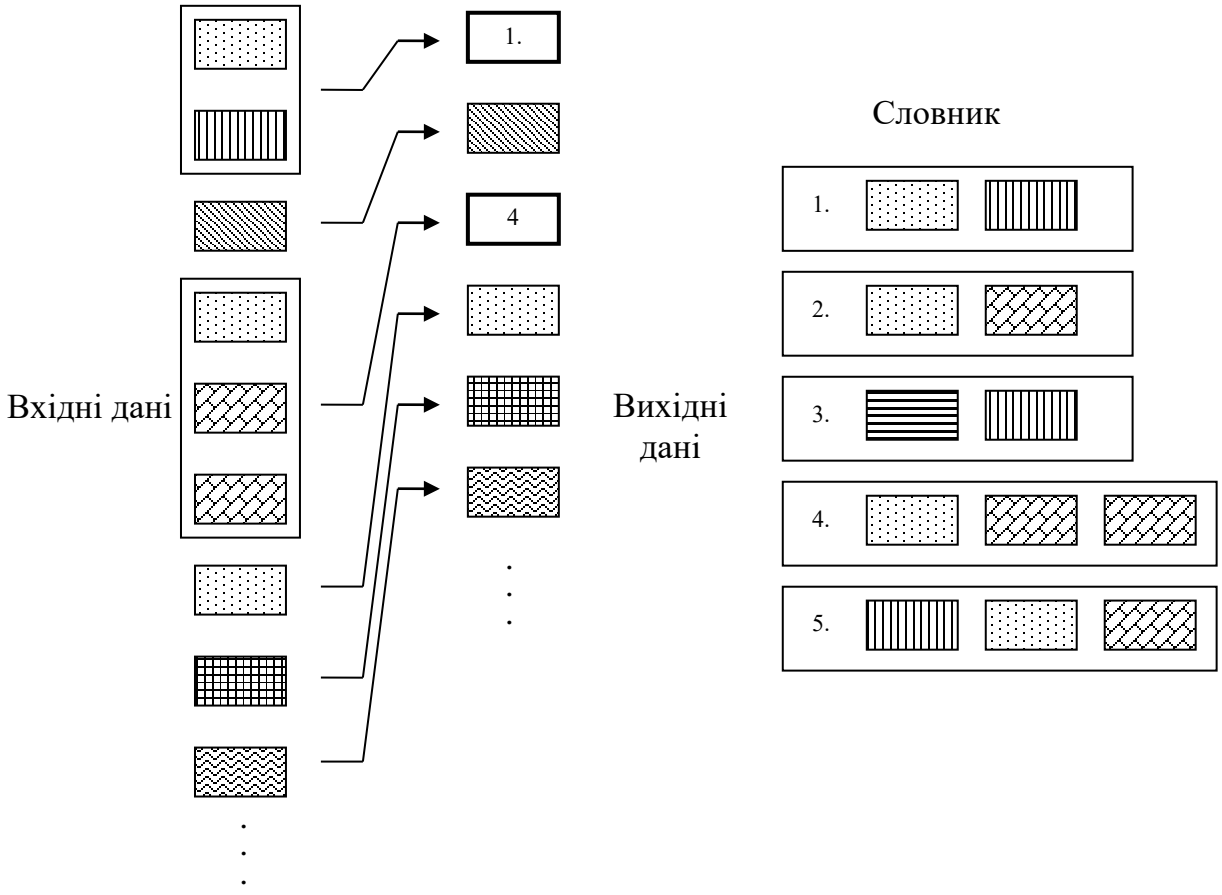


Рис. 2.6

Алгоритми другої групи дещо простіше в поясненні їх роботи, тому почнемо розгляд принципу дії LZ-кодеров з алгоритму LZW.

Розглянемо в найзагальнішому вигляді роботу LZW-кодера і декодера.

2.5.1. Кодування

Процес стискування виглядає досить просто. Ми послідовно прочитуємо символи вхідного потоку (рядок) і перевіряємо, чи є у вже створеній нами таблиці такий рядок.

Якщо рядок є, то прочитуємо наступний символ, а якщо такого рядка немає, - заносимо у вихідний потік код для попереднього знайденого рядка, заносимо її в таблицю і починаємо пошук знову.

Хай на вхід кодера поступає послідовність символів вигляду / WED / WE / WEE / WEB, при цьому розмір алфавіту вхідних символів dim A = 255.

Схема стискування виглядає таким чином:

Вхідні символи	Вихідний код	Нові символи словника
/W	***/***	256 = ***/W***
E	***W***	257 = ***WE***
D	***E***	258 = ***ED***
/	***D***	259 = ***D/***
WE	***256***	260 = ***/WE***
/	***E***	261 = ***E/***
WEE	***260***	262 = ***/WEE***
/W	***261***	263 = ***E/W***
EB	***257***	264 = ***WEB***
<END>	***B***	

В результаті отримаємо вихідний код

/WED<256>E<260><261><257>B.

Як при цьому змінилася довжина вихідної коди порівняно з вхідним ?

Якщо для двійкового кодування рядка / WED / WE / WEE / WEB завдовжки в 15 букв і розміром алфавіту dim A = 255 нам знадобилося б 15 • log2 255 = 15x8 = 120 біт, то для двійкового кодування вихідного рядка кодера / WED <256> E <260> <261> <257> B завдовжки в 10 нових символів з алфавітом в 264 букви – 10 • 9 = 90 біт.

Алгоритм LZW-сжатия виглядає таким чином:

```
set w = NIL
loop
   read a character K
   if wK exists in the dictionary
      w = wK
   else
      output the code for w
      add wK to the string table
      w = K
endloop
```

LZW-кодер починає роботу із словника розміром в 4K, який містить по адресах від 0 до 255 заслання на окремі букви і від 256 до 4095 – заслання на підрядки. В процесі кодування текст піддається розбору на підрядки, де кожен новий підрядок щонайдовший з вже проглянутих плюс один символ. Вона кодується як індекс її префікса плюс додатковий символ, після чого новий підрядок додається в словник і на неї надалі можна буде посилатися.

2.5.2. Декодування

LZW-декодер, обробляючи вхідний потік закодованих даних, відновлює з нього вихідні дані. Так само, як і алгоритм стискування, декодер додає нові рядки в словник всякий раз, коли знаходить у вхідному потоці новий код. Все, що йому залишається зробити, – це перетворити вхідний код у вихідний рядок символів і віддати її на вихід кодера.

Схема роботи LZW-декодера виглядає таким чином:

рядок на вході кодера - /WED<256>E<260><261><257>B.

Вхідні символи	Вихідний рядок	Нові символи словника
/	***/***	
W	***W***	256 = ***/W***
E	***E***	257 = ***WE***
D	***D***	258 = ***ED***
256	***/W***	259 = ***D/***
E	***E***	260 = ***/WE***
260	***/WE***	***261 = E/***
261	***E/***	262 = ***/WEE***
257	***WE***	***263 = E/W***
B	***B***	264 = ***WEB***

Найчудовішою якістю цього способу стискування є те, що весь словник нових символів передається декодеру без власне передачі. В кінці процесу декодування декодер має такий самий словник нових символів, який в процесі кодування був накопичений кодером, при цьому його створення було частиною процесу декодування.

Робота кодера/декодера сімейства LZ77 - першій опублікованій версії LZ-метода - виглядає трохи інакше.

У алгоритмі LZ77 покажчики позначають фрази у вікні постійного розміру, попередні позиції коди. Максимальна довжина замінюваних покажчиками підрядків визначається параметром F (звичайно це від 10 до 20). Ці обмеження дозволяють LZ77 використовувати "ковзаюче вікно" з N символів. З них перші N-F були вже закодовані, а останні F складають попереджуючий буфер.

При кодуванні символу по-перше N-F символах вікна шукають щонайдовший, співпадаючий з цим буфером рядок. Вона може частково перекривати буфер, але не може бути самим буфером.

Знайдена найбільша відповідність потім кодується тріадою [i, j, a] де і є його зсув від початку буфера, j - довжина відповідності, а - перший символ, не відповідний підрядку вікна.

Потім вікно зрушується управо на j +1 символ і готово до нового кроку алгоритму.

Прив'язка певного символу до кожного покажчика гарантує, що кодування виконуватиметься навіть в тому випадку, якщо для першого символу попереджуючого буфера не буде знайдено відповідність.

Об'єм пам'яті, потрібний кодеру і декодеру, обмежується розміром вікна. Кількість біт, необхідна для представлення зсуву (i) в тріаді, складає [log(N-F)]. Кількість символів (j), що замінюються тріадою, може бути закодована [logF] бітами.

Декодування здійснюється дуже просто і швидко. При цьому підтримується той же порядок роботи з вікном, що і при кодуванні, але на відміну від пошуку однакових рядків він, навпаки, копіює їх з вікна відповідно до чергової тріади.

Алгоритми кодування і декодування для LZ77 приведені нижче.

Кодер:

```
while( lookAheadBuffer not empty )
   {
   get a pointer ( position, match ) to the longest match in the window
      for the lookahead buffer;

   if( length > MINIMUM_MATCH_LENGTH )
      {
      output a ( position, length ) pair;
      shift the window length characters along;
      }
   else
      {
      output the first character in the lookahead buffer;
      shift the window 1 character along;
      }
   }
```

Декодер:

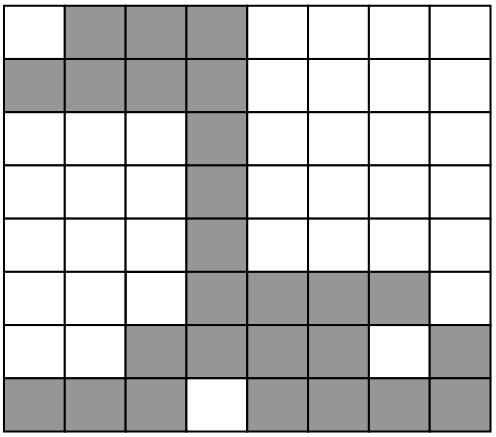
```
Whenever a ( position, length ) pair is
encountered, go to that ( position ) in the window and copy ( length ) bytes
to the output.
```

2.6. Кодування довжин повторень

Кодування довжин ділянок (або повторень) може бути достатнє ефективним при стискуванні двійкових даних, наприклад, чорно-білих факсимільних зображень, чорно-білих зображень, що містять безліч прямих ліній і однорідних ділянок, схем і тому подібне Кодування довжин повторень є одним з елементів відомого алгоритму стискування зображень JPEG.

Ідея стискування даних на основі кодування довжин повторень полягає в тому, що замість кодування власне даних піддаються кодуванню числа, відповідні довжинам ділянок, на яких дані зберігають незмінне значення.

Передбачимо, що потрібно закодувати двійкове (двобарвне) зображення розміром 8 х 8 елементів, приведене на рис. 2.7.



Проскануємо це ...ьорам на зображенні відповідатимуть 0 і 1) ...ий вектор даних

X= (011100001111000 ...'1000111101111101111)

завдовжки 64 біта (швидкість вихідної коди складає 1 біт на елемент зображення).

Виділимо у векторі X ділянки, на яких дані зберігають незмінне значення, і визначимо їх довжини. Результуюча послідовність довжин ділянок - позитивних цілих чисел, відповідних вихідному вектору даних X, - матиме вигляд r = (1, 3, 4, 4, 7, 1, 7, 1, 7, 1, 7, 4, 3, 4, 1, 4, 1, 4).

Тепер цю послідовність, в якій помітна певна повторюваність (одиниць і четвірки значно більше, чим інших символів), можна закодувати яким-небудь статистичним кодом, наприклад, кодом Хаффмена без пам'яті, що має таблицю кодування (таблиця. 2.8)

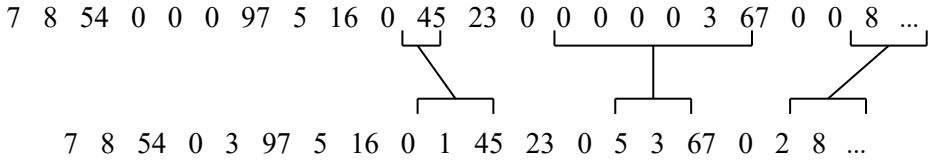
Таблиця 2.8

Кодер	
Довжина участка	Кодове слово
4	0
1	10
7	110
3	111

Для того, щоб вказати, що кодована послідовність починається з нуля, додамо на початку кодового слова префіксний символ 0. В результаті отримаємо кодове слово B (r) = (0100011010110101101011001110100100) завдовжки в 34 біта, тобто результуюча швидкість коди R складе 34/64, або мало чим більше 0,5 біта на елемент зображення. При стискуванні зображень більшого розміру і повторюються елементів, що містять безліч, ефективність стискування може виявитися істотно вищою.

Нижче наведений інший приклад використання кодування довжин повторень, коли в цифрових даних зустрічаються ділянки з великою кількістю нульових значень. Всякий раз, коли в потоці даних зустрічається “нуль”, він кодується двома числами. Перше - 0, таке, що є прапором початку кодування до-

вжини потоку нулів, і друге – кількість нулів в черговій групі. Якщо середнє число нулів в групі більше двох, матиме місце стискування. З іншого боку, велике число окремих нулів може привести навіть до збільшення розміру кодованого файлу:



Ще одним простим і широко використовуваним для стискування зображень і звукових сигналів методом неруйнівного кодування є метод диференціального кодування.

2.7. Диференціальне кодування

Робота диференціального кодера заснована на тому факті, що для багатьох типів даних різниця між сусідніми відліками відносно невелика, навіть якщо самі дані мають великі значення. Наприклад, не можна чекати великої різниці між сусідніми пікселами цифрового зображення.

Наступний простий приклад показує, яка перевага може дати диференціальне кодування (кодування різниці між сусідніми відліками) порівняно з простим кодуванням без пам'яті (кодуванням відліків незалежно один від одного).

Проскануємо 8-бітове (256-рівневе) цифрове зображення, при цьому десять послідовних пікселів мають рівні:

144, 147, 150, 146, 141, 142, 138, 143, 145, 142.

Якщо закодувати ці рівні піксел за пікселом яким-небудь кодом без пам'яті, що використовує 8 біт на піксел зображення, отримаємо кодове слово, що містить 80 біт.

Передбачимо тепер, що перш ніж піддавати відліки зображення кодуванню, ми обчислимо різниці між сусідніми пікселами. Ця процедура дасть нам послідовність наступного вигляду:

144, 147, 150, 146, 141, 142, 138, 143, 145, 142.

⇓ ⇓ ⇓ ⇓ ⇓ ⇓ ⇓ ⇓ ⇓ ⇓

144, 3, 3, - 4, - 5, 1, - 4, 5, 2, -3.

Вихідна послідовність може бути легко відновлена з різницевої простим підсумовуванням (дискретною інтеграцією):

144, 144+3, 147+3, 150–4, 146–5, 141+1, 142–4, 138+5, 143+2, 145-3

⇓ ⇓ ⇓ ⇓ ⇓ ⇓ ⇓ ⇓ ⇓ ⇓

144, 147, 150, 146, 141, 142, 138, 143, 145, 142.

Для кодування першого числа з отриманої послідовності різниць відліків, як і раніше, знадобиться 8 біт, всі останні числа можна закодувати 4-бітовими словами (один знаковий біт і 3 біта на кодування модуля числа).

Таким чином, в результаті кодування отримаємо кодове слово завдовжки 8 + 9*4 = 44 біта або майже удвічі коротше, ніж при індивідуальному кодуванні відліків.

Метод диференціального кодування дуже широко використовується в тих випадках, коли природа даних така, що їх сусідні значення трохи відрізняються один від одного, при тому, що самі значення можуть бути скільки завгодно великими.

Це відноситься до звукових сигналів, особливо до мови, зображень, сусідні пікселі яких мають практично однакові яскравості і колір і тому подібне В той же час цей метод абсолютно не личить для кодування текстів, креслень або яких-небудь цифрових даних з незалежними сусідніми значеннями.

2.8. Методи стискування з втратою інформації

Як вже раніше наголошувалося, існують два типи систем стискування даних:

- без втрат інформації (неруйнівні);
- з втратами інформації (руйнівні).

При неруйнівному кодуванні вихідні дані можуть бути відновлені із стислих в первинному вигляді, тобто абсолютно точно. Таке кодування застосовується для стискування текстів, баз даних, комп'ютерних програм і даних і тому подібне, де недопустимо їх навіть щонайменша відмінність. Всі розглянуті вище методи кодування відносилися саме до неруйнівних.

На жаль, неруйнівне стискування, при всій привабливості перспективи отримати абсолютний збіг вихідних і відновлених даних, має невисоку ефективність – коефіцієнти неруйнівного стискування рідко перевищують 3.5 (за винятком випадків кодування даних з високою мірою повторюваності однакових ділянок і тому подібне).

В той же час дуже часто немає необхідності в абсолютній точності передачі вихідних даних споживачеві. По-перше, самі джерела даних володіють обмеженим динамічним діапазоном і виробляють вихідні повідомлення з певним рівнем спотворень і помилок. Цей рівень може бути великим або меншим, але абсолютної точності відтворення досягти неможливо.

По-друге, передача даних по каналах зв'язку і їх зберігання завжди виробляються за наявності різного роду перешкод. Тому прийняте (відтворене) повідомлення завжди певною мірою відрізняється від переданого, тобто на практиці неможлива абсолютно точна передача за наявності перешкод в каналі зв'язку (у системі зберігання).

Нарешті, повідомлення передаються і зберігаються для їх сприйняття і використання одержувачем. Одержувачі ж інформації - органи чуття людини, виконавчі механізми і так далі - також володіють кінцевою роздільною здатністю, тобто не помічають незначної різниці між абсолютно точним і наближеним значеннями відтворного повідомлення. Поріг чутливості до спотворень також може бути різним, але він завжди є.

Кодування з руйнуванням враховує ці аргументи на користь наближеного відновлення даних і дозволяє отримати за рахунок деякої контрольованої по величині помилки коефіцієнти стискування, інколи в десятки разів що перевищують міру стискування для неруйнівних методів.

Більшість методів руйнівного стискування заснована на кодуванні не самих даних, а деяких лінійних перетворень від них, наприклад, коефіцієнтів дискретного перетворення Фур'є (ДПФ), коефіцієнтів косинусного перетворення, перетворень Хаару, Уолша і тому подібне

Для того, щоб зрозуміти, на чому заснована висока ефективність систем руйнівного стискування і чому кодування перетворень виявляється значно ефективнішим, ніж кодування вихідних даних, розглянемо як приклад популярний метод стискування зображень JPEG (“джипег”), який містить в собі всі необхідні атрибути системи руйнівного стискування. Зараз не вдаватимемося до формульные нетрів, головне – зрозуміти ідею кодування перетворень.

Потрібно буде також розглянути і єство самих лінійних перетворень, вживаних для стискування, оскільки без розуміння їх фізичного сенсу важко з'ясувати причини отримуваних ефектів.

2.8.1. Кодування перетворень. Стандарт стискування JPEG

Популярний стандарт кодування зображень JPEG (Joint Photographers Experts Group) є дуже хорошою ілюстрацією до пояснення принципів руйнівного стискування на основі кодування перетворень.

Основну ідею кодування перетворень можна зрозуміти з наступних простих міркувань. Допустимо, ми маємо справу з деяким цифровим сигналом (послідовністю відліків Котельникова). Якщо відкинути в кожному з відліків половину двійкових розрядів (наприклад, 4 розряди з восьми), то удвічі зменшиться швидкість коди R і загубиться половина інформації, що міститься в сигналі.

Якщо ж піддати сигнал перетворенню Фур'є (або якому-небудь іншому подібному лінійному перетворенню), розділити його дві складові – НЧ і ВЧ, продискретизовать, піддати квантуванню кожну з них і відкинути половину двійкових розрядів лише у високочастотній складовій сигналу, то результуюча швидкість коди зменшиться на одну третину, а втрата інформації складе всього 5%.

Цей ефект обумовлений тим, що низькочастотні складові більшості сигналів (крупні деталі) зазвичай набагато інтенсивніші і несуть значно більше інформації, ніж високочастотні складові (дрібні деталі). Це в рівній мірі відноситься і до звукових сигналів, і до зображень.

Розглянемо роботу алгоритму стискування JPEG при кодуванні чорно-білого зображення, представленого набором своїх відліків (пікселів) з числом градацій яскравості в 256 рівнів (8 двійкових розрядів). Це найпоширеніший спосіб зберігання зображень - кожній крапці на екрані відповідає один байт (8 біт - 256 можливих значень), що визначає її яскравість. 255 - яскравість максимальна (білий колір), 0 - мінімальна (чорний колір). Проміжні значення складають всю останню гамму сірих кольорів (рис.2.8).

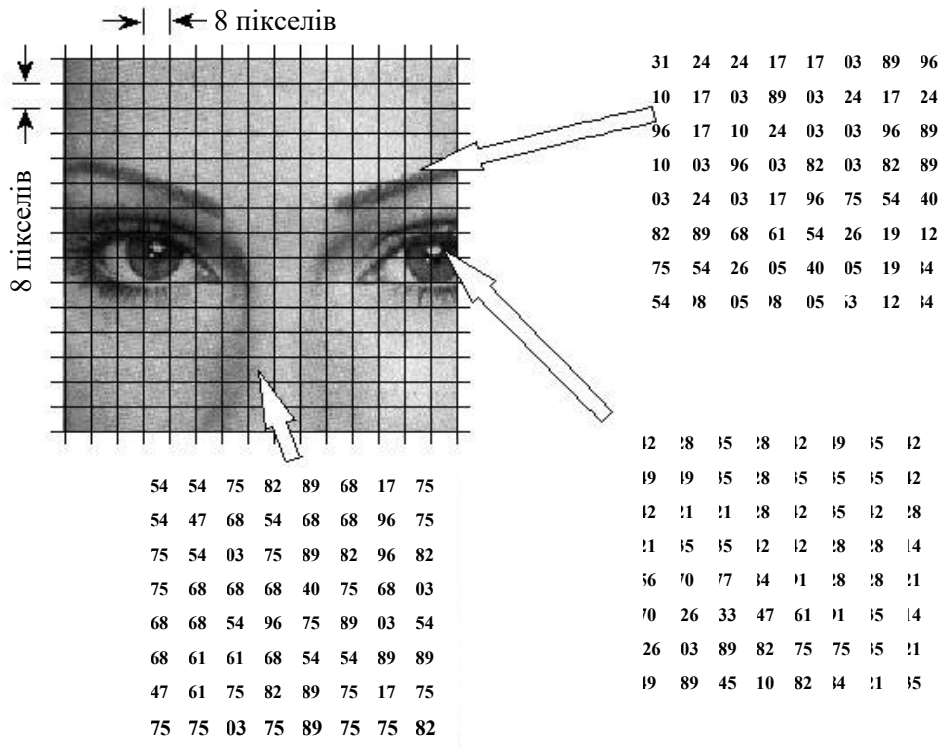


Рис. 2.8

Робота алгоритму стискування JPEG починається з розбиття зображення на квадратні блоки розміром 8x8 = 64 пікселі. Чому саме 8x8, а не 2x2 або 32x32 ? Вибір такого розміру блоку обумовлений тим, що при його малому розмірі ефект кодування буде невеликим (при розмірі 1x1 – взагалі відсутній), а при великому - властивості зображення в межах блоку сильно змінюватимуться і ефективність кодування знову знизиться.

На рис.2.8 змальовано декілька таких блоків (у вигляді матриць цифрових відліків), узятих з різних ділянок зображення. Надалі ці блоки оброблятимуться і кодуватимуться незалежно один від одного.

Другий етап стискування – застосування до всіх блоків дискретного косинусного перетворення – DCT. Для стискування даних намагалися застосувати безліч різних перетворень, у тому числі спеціально розроблених для цих цілей, наприклад, перетворення Карунены-Лоэва, що забезпечує максимально можливий коефіцієнт стискування. Але воно дуже складно реалізується на практиці. Перетворення Фур'є виконується дуже просто, але не забезпечує хорошого стискування. Вибір був зупинений на дискретному косинусному перетворенні, що виявляє різновидом ПФ. На відміну від ПФ, яке застосовує для розкладання сигналу синусні і косинусні частотні складові, в DCT використовуються лише косинусні складові. Дискретне косинусне перетворення дозволяє перейти від просторового представлення зображення (набором відліків або пікселів) до спектральної вистави (набором частотних складових) і навпаки.

Дискретне косинусне перетворення від зображення IMG (x, в) може бути записане таким чином:

$$DCT(u,v) = sqrt(2/N) \sum_{i,j} IMG(x_i, y_j) cos((2i+1)\pi u/2N) cos((2j+1)\pi v/2N), \quad (2.17)$$

где $N = 8,\ 0 < i < 7,\ 0 < j < 7,$

або ж, в матричній формі

$$RES = DCT^T * IMG * DCT, \quad (2.18)$$

де DCT - матриця базисних (косинусних) коефіцієнтів для перетворення розміром 8x8, що має вигляд:

$$DCT = \begin{vmatrix} .353553 & .353553 & .353553 & .353553 & .353553 & .353553 & .353553 & .353553 \\ .490393 & .415818 & .277992 & .097887 & -.097106 & -.277329 & -.415375 & -.490246 \\ .461978 & .191618 & -.190882 & -.461673 & -.462282 & -.192353 & .190145 & .461366 \\ .414818 & -.097106 & -.490246 & -.278653 & .276667 & .490710 & .099448 & -.414486 \\ .353694 & -.353131 & -.354256 & .352567 & .354819 & -.352001 & -.355378 & .351435 \\ .277992 & -.490246 & .096324 & .416700 & -.414486 & -.100228 & .491013 & -.274673 \\ .191618 & -.462282 & .461366 & -.189409 & -.193822 & .463187 & -.460440 & .187195 \\ .097887 & -.278653 & .416700 & -.490862 & .489771 & -.413593 & .274008 & -.092414 \end{vmatrix} \quad (2.19)$$

Отже, в результаті застосування до блоку зображення розміром 8x8 пікселів дискретного косинусного перетворення отримаємо двовимірний спектр, що також має розмір 8x8 відліків. Іншими словами, 64 числа, що представляють відліки зображення, перетворяться на 64 числа, що представляють відліки його DCT-спектра.

А тепер нагадаємо, що таке спектр сигналу. Це – величини коефіцієнтів, з якими відповідні спектральні складові входять в суму, яка в результаті і дає цей сигнал. Окремі спектральні складові, на які розкладається сигнал, часто називають базисними функціями. Для ПФ базисними функціями є синуси і косинуси різних частот.

Для 8x8 DCT система базисних функцій задається формулою

$$b[x,y] = cos\left[\frac{(2x+1)u\pi}{16}\right] cos\left[\frac{(2y+1)v\pi}{16}\right], \quad (2.20)$$

а самі базисні функції виглядають подібно до приведених на рис.2.9.

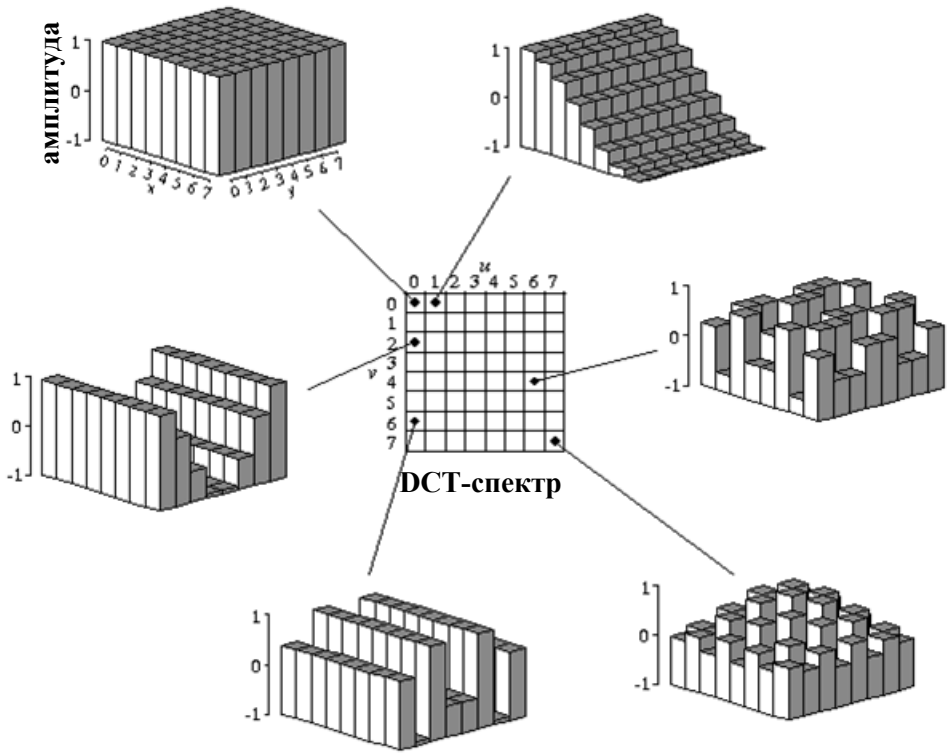


Рис. 2.9

Сама низькочастотна базисна функція, відповідна індексам (0,0), змальована в лівому верхньому кутку малюнка, сама високочастотна – в нижнем правом. Базисна функція для (0,1) є половиною періоду косинусоїди по одній координаті і константою - по іншій, базисна функція з індексами (1,0) – те ж саме, але повернена на 900 .

Дискретне косинусне перетворення обчислюється шляхом поелементного перемножування і підсумовування блоків зображення 8x8 пікселів з кожною з цих базисних функцій. В результаті, наприклад, компонента DCT-спектра з індексами (0,0) буде просто сумою всіх елементів блоку зображення, тобто середню для блоку яскравість. У компоненту з індексом (0,1) усереднюються з однаковими вагами всі горизонтальні деталі зображення, а по вертикалі найбільша вага привласнюється елементам верхньої частини зображення і так далі Можна відмітити, що чим нижче і правіше в матриці DCT його компонента, тим більше високочастотним деталям зображення вона відповідає.

Для того, щоб отримати вихідне зображення по його DCT-спектру (виконати зворотне перетворення), потрібно тепер базисну функцію з індексами (0,0) помножити на спектральну компоненту з координатами (0,0), додати до результату твір базисної функції (1,0) на спектральну компоненту (1,0) і так далі

У приведеній нижче таблиці. 2.9 видно числові значення одного з блоків зображення і його DCT-спектра:

Таблиця 2.9

Вихідні дані							
139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	161	160	160	159	159	159
159	160	161	162	162	155	155	155
161	161	161	161	160	157	157	157
161	162	161	163	162	157	157	157
162	162	161	161	163	158	158	15
Результат DCT							
235,6	-1	-12,1	-5,2	2,1	-1,7	-2,7	1,3
-22,6	-17,5	-6,2	-3,2	-2,9	-0,1	0,4	-1,2
-10,9	-9,3	-1,6	1,5	0,2	-0,9	-0,6	-0,1
-7,1	-1,9	0,2	1,5	0,9	-0,1	0	0,3
-0,6	-0,8	1,5	1,6	-0,1	-0,7	0,6	1,3
1,8	-0,2	1,6	-0,3	-0,8	1,5	1	-1
-1,3	-0,4	-0,3	-1,5	-0,5	1,7	1,1	-0,8
-2,6	1,6	-3,8	-1,8	1,9	1,2	-0,6	-0,4

Відзначимо дуже цікаву особливість отриманого DCT-спектра: найбільші його значення зосереджені в лівому верхньому кутку таблиці. 2.9 (низькочастотні складові), права ж нижня його частина (високочастотні складові) заповнена відносно невеликими числами. Чисел цих, правда, стільки ж, як і в блоці зображення: 8x8 = 64, тобто доки жодного стискування не сталося, і, якщо виконати зворотне перетворення, отримаємо той же самий блок зображення.

Наступним етапом роботи алгоритму JPEG є квантування (таблиця. 2.10).

Якщо уважно поглянути на отриманих в результаті DCT коефіцієнти, то буде видно, що добра їх половина - нульові або має дуже невеликі значення (1 - 2). Це високі частоти, які (зазвичай) можуть бути безболісно відкинуті або, принаймні, закруглені до найближчого цілого значення.

Квантування полягає в діленні кожного коефіцієнта DCT на деяке число відповідно до матриці квантування. Ця матриця може бути фіксованою або, для якіснішого і ефективнішого стискування, отримана в результаті аналізу характеру вихідної картинки. Чим більше числа, на яких відбувається ділення, тим більше в результаті ділення буде нульових значень, а значить, сильніше стискування і помітніше за втрату.

Таблиця 2.10

Раніше отриманий результат DCT							
235,6	-1	-12,1	-5,2	2,1	-1,7	-2,7	1,3
-22,6	-17,5	-6,2	-3,2	-2,9	-0,1	0,4	-1,2
-10,9	-9,3	-1,6	1,5	0,2	-0,9	-0,6	-0,1
-7,1	-1,9	0,2	1,5	0,9	-0,1	0	0,3
-0,6	-0,8	1,5	1,6	-0,1	-0,7	0,6	1,3
1,8	-0,2	1,6	-0,3	-0,8	1,5	1	-1
-1,3	-0,4	-0,3	-1,5	-0,5	1,7	1,1	-0,8
-2,6	1,6	-3,8	-1,8	1,9	1,2	-0,6	-0,4
Таблиця квантування							
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99
Результат квантування							
15	0	-1	0	0	0	0	0
-2	-1	0	0	0	0	0	0
-1	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Абсолютно вочевидь, що від вибору таблиці квантування в значній мірі залежатиме як ефективність стискування – число нулів в квантованому (закругленому) спектрі, – так і якість відновленої картинки.

Таким чином, ми округлили результат DCT і отримали більшою чи меншою мірою спотворений поблочний спектр зображення.

Наступним етапом роботи алгоритму JPEG є перетворення 8x8 матриць DCT-спектра в лінійну послідовність. Але робиться це так, щоб згрупувати по можливості разом всі великі значення і всі нульові значення спектру. Абсо-

лютно вочевидь, що для цього потрібно прочитати елементи матриці коефіцієнтів DCT в порядку, змальованому на рис.2.10, тобто зигзагоподібно - з лівого верхнього кута до правого нижнему. Ця процедура називається зигзаг-скануванням.

В результаті такого перетворення квадратна матриця 8x8 квантованих коефіцієнтів DCT перетвориться на лінійну послідовність з 64 чисел, велика частина з яких – це що йдуть підряд нулі. Відомо, що такі потоки можна дуже ефективно стискувати шляхом код Саме так це і робиться.

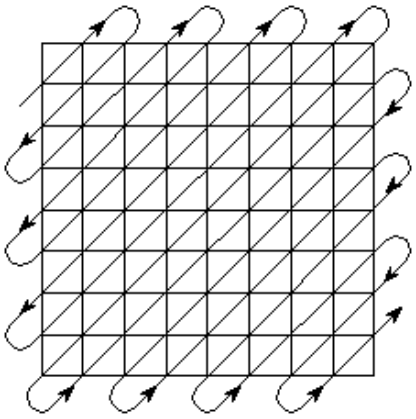


Рис. 2.10

На наступному, п'ятому етапі JPEG-кодування ланцюжки нулів, що вийшли, піддаються кодуванню довжин повторень.

І, нарешті, останнім етапом роботи алгоритму JPEG є кодування послідовності, що вийшла, яким-небудь статистичним алгоритмом. Зазвичай використовується арифметичне кодування або алгоритм Хаффмена. В результаті виходить нова послідовність, розмір якої істотно менше розміру масиву вихідних даних.

Останні два етапи кодування зазвичай називають вторинним стискуванням, і саме тут відбувається неруйнівне статистичне кодування, і з врахуванням характерної структури даних - істотне зменшення їх об'єму.

Декодування даних стислих згідно алгоритму JPEG виробляється точно так, як і кодування, але всі операції слідують в зворотному порядку.

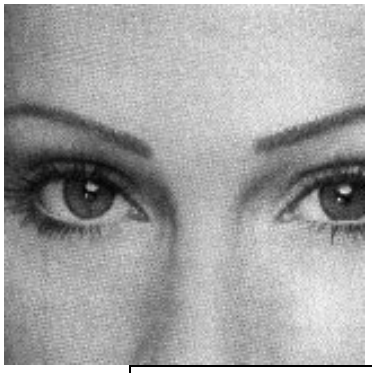
Після неруйнівного розпаковування методом Хаффмена (або LZW, або арифметичного кодування) і розставляння лінійної послідовності в блоки розміром 8x8 числами спектральні компоненты деквантуются за допомогою збережених при кодуванні таблиць квантування. Для цього розпаковані 64 значення DCT умножаються на відповідні числа з таблиці. Після цього кожен блок піддається зворотному косинусному перетворенню, процедура якого збігається з прямим і розрізняється лише знаками в матриці перетворення. Послідовність

дій при декодуванні і отриманий результат ілюструються приведеній нижче таблиці. 2.11.

Таблиця 2.11

Квантовані дані							
15		1					0
-2	1						0
-1	1						0
0							0
0							0
0							0
0							0
0							0
Деквантованниє дані							
240		10					0
-24	12						0
-14	13						0
0							0
0							0
0							0
0							0
0							0
Результат зворотного DCT							
144	46	49	52	54	56	56	1 56
148	50	52	54	56	56	56	1 56
155	56	57	58	58	57	56	1 55
160	61	61	62	61	59	57	1 55
163	63	64	63	62	60	58	1 56
163	64	64	64	62	60	58	1 57
160	61	62	62	62	61	59	1 58

158	59	61	61	62	61	59	58 1
Для порівняння - вихідні дані							
139	44	49	53	55	55	55	55
144	51	53	56	59	56	56	56



150	55	60	63	58	56	56	56
159	61	61	60	60	59	59	59
159	60	61	62	62	55	55	55
161	61	61	61	60	57	57	57
161	62	61	63	62	57	57	57
162	62	61	61	63	58	58	5

Вочевидь, що відновлені дані декілька відрізняються від початкових. Це природно, тому що JPEG і розроблявся, як стискування з втратами.

На представленому нижче рис.2.11 приведено вихідне зображення (справа), а також зображення, стисле з використанням алгоритму JPEG в 10 разів (зліва) і в 45 разів (у центрі). Втрата якості в останньому випадку сповна помітна, але і виграш за об'ємом теж очевидний.

Рис. 2.11

Отже, JPEG-сжатие складається з наступних етапів:

1. Розбиття зображення на блоки розміром 8x8 пікселами.
2. Застосування до кожного з блоків дискретного косинусного перетворення.
3. Округлення коефіцієнтів DCT відповідно до заданої матриці вагових коефіцієнтів.
4. Перетворення матриці закруглених коефіцієнтів DCT в лінійну послідовність шляхом їх зигзагоподібного читання.
5. Кодування повторень для скорочення числа нульових компнент.
6. Статистичне кодування результату кодом Хаффмена або арифметичним кодом.

Декодування виробляється так само, але в зворотному порядку.

Істотними позитивними сторонами алгоритму стискування JPEG є:

- можливість завдання в широких межах (від 2 до 200) міри сжа- тия;
- можливість роботи із зображеннями будь-якої розрядності;
- симетричність процедур стискування – розпаковування.

До недоліків можна віднести наявність ореолу на різких переходах кольорів - ефект Гіббса, а також розпад зображення на окремі квадратики 8x8 при високій мірі стискування.

2.8.2. Фрактальний метод

Фрактальне стискування засноване на тому, що зображення представляється в компактнішій формі за допомогою коефіцієнтів системи итерируемых функцій (Iterated Function System — IFS).

Перш ніж розглядати сам процес фрактального стискування, розберемося, як IFS будує зображення, тобто розглянемо процес декомпресії.

Строго кажучи, IFS є набором тривимірних аффинных перетворень, в нашому випадку тих, що переводять одне зображення в інше. Перетворенню піддаються крапки в тривимірному просторі (координата точки зображення X, координата точки зображення Y і яскравість крапки I).

Спрощено цей процес можна пояснити таким чином. Розглянемо так звану машину (рис.2.12), що фотокопіює, складається з екрану, на якому змальована вихідна картинка, і системи лінз, що проектують зображення на інший екран. Машина, що фотокопіює, може виконувати наступні дії:

- лінзи можуть проектувати частину зображення довільної форми в будь-яке інше місце нового зображення;
- області, в які проектуються зображення, не перетинаються;
- лінза може міняти яскравість і зменшувати контрастність;
- лінза може дзеркально відображати і повертати свій фрагмент зображення;
- лінза повинна масштабувати (зменшувати) свій фрагмент зображення.

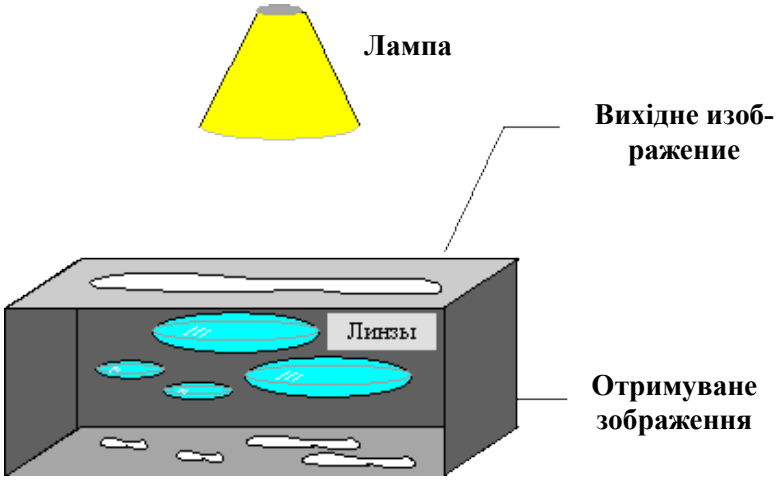


Рис. 2.12

Розставляючи лінзи і міняючи їх характеристики, можна управляти отримуваним зображенням. Одна ітерація роботи машини полягає в тому, що по вихідному зображенню за допомогою проектування будується нове, після чого нове береться як початковий. Затверджується, що в процесі ітерацій ми отримаємо зображення, яке перестане змінюватися. Воно залежатиме лише від розташування і характеристик лінз і не залежатиме від вихідної картинки. Це зображення називається “Нерухомою крапкою”, або аттрактором даної IFS. Відповідна теорія гарантує наявність рівно однієї нерухомої крапки для кожної IFS.

Оскільки відображення лінз є таким, що стискує, кожна лінза в явному вигляді задає самоподобные області в нашому зображенні. Завдяки самоподобию отримуємо складну структуру зображення при будь-якому збільшенні. Таким чином, інтуїтивно зрозуміло, що система итерируемых функцій задає фрактал — самоподобный математичний об'єкт.

Найбільш відомим із зображень, отриманих за допомогою IFS, є так звана “папороть Барнслі” (рис.2.13), що задається чотирма аффинными перетвореннями (або, в нашій термінології, “лінзами”). Кожне перетворення кодується

буквально ліченими байтами, тоді як зображення, побудоване з їх допомогою, може займати і декілька мегабайт.

Рис. 2.13

На цьому зображенні можна виділити 4 області, об'єднання яких покриває все зображення і кожна з яких подібна до всього зображення (не забувайте про стебло папороті).
З вищесказаного стає зрозуміло, як працює фрактальний кодер, і також вочевидь, що для стискування йому знадобиться дуже багато часу.
Фактично фрактальна компресія — це пошук самоподобных областей в зображенні і визначення для них параметрів аффинных перетворень. Для цього потрібно буде виконати перебір і порівняння всіх можливих фрагментів зображення різного розміру. Навіть для невеликих зображень при обліку дискретності ми отримаємо астрономічне число перебираних варіантів, причому навіть різке звуження класів перетворень, наприклад, за рахунок масштабування, лише в певну кількість разів не дає помітного виграшу в часі. Крім того, при цьому втрачається якість зображення. Переважна більшість досліджень в області фрактального стискування зараз направлені на зменшення часу архівації, необхідного для здобуття якісного зображення.

2.8.3. Рекурсивний (хвилевий) алгоритм

Англійська назва рекурсивного стискування — wavelet. На російську мову воно переводиться як хвилеве стискування і як стискування з використанням сплесків. Цей вигляд архівації відомий досить давно і безпосередньо виходить з ідеї використання когерентності областей. Орієнтований алгоритм на кольорові і чорно-білі зображення з плавними переходами, ідеальний для картинок типа рентгенівських знімків. Коефіцієнт стискування задається і варіюється в межах 5 - 100. При спробі задати більший коефіцієнт на різких кордонах, особливо проходящих по діагоналі, виявляється “сходовий ефект” — сходинки різної яскравості розміром в декілька пікселів.

Ідея алгоритму полягає в тому, що замість кодування власне зображень зберігається різниця між середніми значеннями сусідніх блоків в зображенні, яка зазвичай набуває значень, близьких до 0.

Так, два числа a2i і a2i+1 завжди можна представити у вигляді b1i= =(a2i+a2i+1)/2 і b2i=(a2i-a2i+1) /2. Аналогічно послідовність ai може бути попарно переведена в послідовність b1,2i.

Розглянемо приклад. Хай ми стискуємо рядок з восьми значень яскравості пікселів (ai): (220, 211, 212, 218, 217, 214, 210, 202). Отримаємо наступні послідовності b1i, і b2i: (215.5, 215, 215.5, 206) і (4.5, -3, 1.5, 4). Відмітимо, що значення b2i досить близькі до 0. Повторимо операцію, розглядаючи b1i як a і. Дана дія виконується як би рекурсивно, звідки і назва алгоритму. З (215.5, 215, 215.5, 206) отримаємо (215.25, 210.75) (0.25, 4.75). Отримані коефіцієнти, округливши до цілих і стискує, наприклад, за допомогою алгоритму Хаффмана, можна вважати результатом кодування. Відмітимо, що ми застосовували наше перетворення до ланцюжка лише двічі. Реально можна дозволити собі вживання wavelet-преобразования 4-6 разів, що дозволить досягти помітних коефіцієнтів стискування.

Алгоритм для двовимірних даних реалізується аналогічно.

Якщо у нас є квадрат з чотирьох крапок з яркостями $a_{2i,2j}$, a_{2i+1}, $a_{2j,}$ a_{2i}, $_{2j+1}$, и a_{2i+1}, a_{2j+1}, то

$$
\begin{aligned}
b^1_{i,j} &= (a_{2i,2j} + a_{2i+1,2j} + a_{2i,2j+1} + a_{2i+1,2j+1})/4,\\
b^2_{i,j} &= (a_{2i,2j} + a_{2i+1,2j} - a_{2i,2j+1} - a_{2i+1,2j+1})/4,\\
b^2_{i,j} &= (a_{2i,2j} - a_{2i+1,2j} + a_{2i,2j+1} - a_{2i+1,2j+1})/4,\\
b^3_{i,j} &= (a_{2i,2j} - a_{2i+1,2j} - a_{2i,2j+1} + a_{2i+1,2j+1})/4.
\end{aligned}
\qquad (2.21)
$$

Використовуючи ці формули, для зображення 512x512 пікселі отримаємо після першого перетворення вже 4 матриці розміром 256x256 елементами (рис. 2.14, 2.15)

Исходное изображение	B^1	B^2
	B^3	B^4

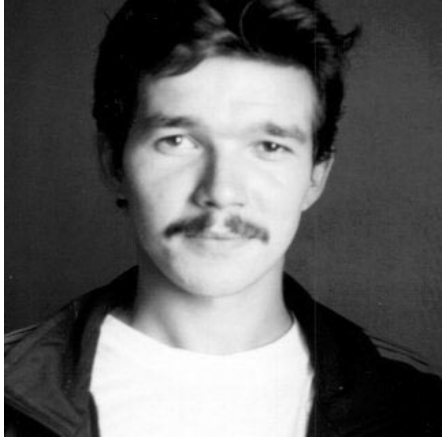


Рис. 2.14

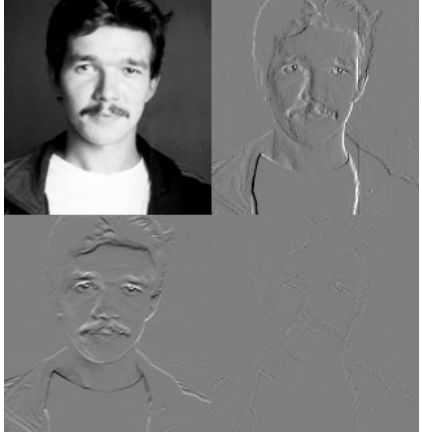


Рис. 2.15

У першій, як легко здогадатися, зберігається зменшена копія зображення, в другій — усереднені різниці пар значень пікселів по горизонталі, в третій — усереднені різниці пар значень пікселів по вертикалі, в четвертій — усереднені різниці значень пікселів по діагоналі.

По аналогії з двовимірним випадком можна повторити перетворення і отримати замість першої матриці 4 матриці розміром 128x128.

Повторивши перетворення втретє, отримаємо у результаті 4 матриці 64x64, 3 матриці 128x128 і 3 матриці 256x256. Подальше стискування відбувається за рахунок того, що в різницевих матрицях є велике число нульових або близьких до нуля значень, які після квантування ефективно стискуються.

До достоїнств цього алгоритму можна віднести те, що він дуже легко дозволяє реалізувати можливість поступового "прояву" зображення при передачі зображення по мережі. Крім того, оскільки на початку зображення ми фактично зберігаємо його зменшену копію, спрощується показ "огрубленого" зображення по заголовку.

На відміну від JPEG і фрактального алгоритму даний метод не оперує блоками, наприклад 8x8 пікселів. Точніше, ми оперуємо блоками 2x2, 4x4, 8x8 і так далі Проте за рахунок того, що коефіцієнти для цих блоків зберігаються незалежно, можна досить легкий уникнути дроблення зображення на "мозаїчні" квадрати.

2.9. Методи стискування рухливих зображень (відео)

Основною проблемою в роботі з рухливими зображеннями є великі об'єми даних, з якими доводиться мати справу. Наприклад, при записі на компакт-диск в середній якості на нього можна помістити декілька тисяч фотографій, більше 10 годин музики і все півгодини відео. Відео телевізійного формату – 720x576 крапок і 25 кадрів в секунду в системі RGB - вимагає потоку даних приблизно 240 Мбіт/с (1,8 Гбіт/мін). При цьому звичайні методи стискування, орієнтовані на кодування окремих кадрів (у тому числі і JPEG), не рятують положення, оскільки навіть при зменшенні бітового потоку в 10 - 20 разів він залишається занадто великим для практичного використання.

При стискуванні рухливих зображень враховується наявність в них декількох типів надмірності:

- *когерентність (однобарвність) областей зображення – незначна зміна кольору зображення в його сусідніх пікселах; це властивість зображення використовується при його руйнівному стискуванні всіма відомими методами;*
- *надмірність в колірних площинах, що відображає високу міру зв'язку інтенсивностей різних колірних компонент зображення і важливість його яскравості компоненты;*
- *подібність між кадрами – використання того факту, що при швидкості 25 кадрів в секунду (а це мінімальна їх частота, при якій непомітне мигтіння зображення, пов'язане із зміною кадрів) відмінність в сусідніх кадрах дуже трохи.*

З середини 80-х рр. багато західних університетів і лабораторії фірм працювали над створенням алгоритму компресії цифрового відеосигналу. З'явилося чимале число внутріфірмових стандартів. Область ця дуже специфічна і динамічна - міжнародні стандарти з'являються буквально через 2-3 роки після створення алгоритму. Розглянемо існуючі стандарти в області цифрового відео.

У 1988 році в рамках Міжнародної організації по стандартизації (ISO) почала роботу група MPEG (Moving Pictures Experts Group) - група експертів в області цифрового відео. У вересні 1990 року був представлений попередній стандарт кодування MPEG-I. У січні 1992 року робота над MPEG-I була завершена.

Робота ця була почата не на порожньому місці, і як алгоритм MPEG має декілька попередників. Це перш за все JPEG - універсальний алгоритм, призначений для стискування статичних повнокольорових зображень. Його універсальність означає, що алгоритм дає непогані результати на широкому класі зображень. Алгоритм використовує конвеєр з декількох перетворень. Ключовим є дискретне косинусне перетворення (ДКП), що дозволяє в широких межах міняти міру стискування без помітної втрати якості зображення. Остання фраза означає, що розрізнити на око відновлене і вихідне зображення практично неможливо. Ідея алгоритму полягає в тому, що в реальних зображеннях малі амплітуди високих частот при розкладанні матриці зображення в подвійний ряд по косинусах. Можна досить вільний огрублювати їх виставу, не сильно погіршуючи зображення. Крім того, використовується переклад в інший колірний простір (YUV), групове кодування і кодування Хаффмана.

Алгоритм стискування

Технологія стискування відео в MPEG розпадається на дві частини: зменшення надмірності відеоінформації в тимчасовому вимірі, засноване на тому, що сусідні кадри, як правило, відрізняються не сильно, і стискування окремих зображень.

Зменшення надмірності в часовому вимірі

Аби задовольнити суперечливим вимогам і збільшити гнучкість алгоритму, в послідовності кадрів, складових рухливе зображення, виділяють чотирьох типів кадрів:

- ***I-кадры - незалежно стислі (I-Intrapictures);***
- ***P-кадры - стислі з використанням заслання на одне зображення (P-Predicted);***
- ***B-кадри - стислі з використанням заслання на два зображення (B-bidirection);***

- ***Вс-кадры - незалежно стислі з великою втратою якості (використовуються лише при швидкому пошуку).***

І-кадры забезпечують можливість довільного доступу до будь-якого кадру, будучи своєрідними вхідними крапками в потік даних для декодера.

Р-кадры використовують при архівації заслання на один І- або Р-кадр, підвищуючи тим самим міру стискування фільму в цілому.

В-кадри, використовуючи заслання на два кадри, що знаходяться попереду і позаду, забезпечують найвищу міру стискування; самі як заслання використовуватися не можуть.

Послідовність кадрів у фільмі може бути, наприклад, такий: I B B P B B P B B P B B P B B I B B P ...., або I P B P B P B I P B P B ...

Частоту І-кадра вибирають виходячи з вимог на час довільного доступу і надійності потоку при передачі по каналу з перешкодами, співвідношення між Р і В-кадрами – виходячи з необхідної міри стискування і складності декодера, оскільки для того, щоб розпакувати В-кадр, потрібно вже мати як передуючий, так і наступний за ним кадри.

Одне з основних понять при стискуванні декількох зображень - макроблок. Макроблок - це матриця пікселів 16x16 елементів (розмір зображення має бути кратний 16). Така величина вибрана не випадково - ДКП працює з матрицями розміром 8x8 елементами. При стискуванні кожен макроблок з колірного простору RGB переводиться в колірний простір YUV. Матриця, відповідна Y (компоненту яскравості), перетворюється на чотири вихідні матриці для ДКП, а матриці, відповідні компонентам U і V, проріджуються на всі парні рядки і стовпці, перетворюючись на одну матрицю для ДКП.

Таким чином, ми відразу отримуємо стискування в два рази, користуючись тим, що око людини гірше розрізняє колір окремої точки зображення, чим її яскравість.

Окремі макроблоки стискуються незалежно, тобто у в-кадрах можна стискувати макроблок як І-блок, Р-блок із засланням на попередній кадр, Р-блок із засланням на подальший кадр і, нарешті, як в-блок.

Стискування окремих кадрів

Існує досить багато алгоритмів, що стискують статичні зображення. З них найчастіше використовуються алгоритми на базі дискретного косинусного перетворення. Алгоритм стискування окремих кадрів в MPEG схожий на відповідний алгоритм для статичних зображень - JPEG. Нагадаємо, як виглядає процедура JPEG -кодирования.

До макроблоків, які готує алгоритм зменшення надмірності в тимчасовому вимірі, застосовується ДКП. Само перетворення полягає в розкладанні значень дискретної функції два змінних в подвійний ряд по косинусах деяких частот.

Дискретне косинусне перетворення переводить матрицю значень яркостей в матрицю амплітуд спектральних компонент, при цьому амплітуди, відповідні нижчим частотам, записуються в лівий верхній кут матриці, а ті, які відповідають вищим, - в правий нижний. Оскільки в реалістичних зображеннях високочастотна складова дуже мала по амплітуді, в результуючій матриці значення під побічною діагоналлю або близькі, або дорівнюють нулю.

До отриманої матриці амплітуд застосовується операція квантування. Саме на етапі квантування - групового кодування - в основному і відбувається адаптивне стискування, і тут же виникають основні втрати якості фільму. Квантування - це цілочисельне поелементне ділення матриці амплітуд на матрицю квантування (МК). Підбір значень МК дозволяє збільшувати або зменшувати втрати по певних частотах і регулювати якість зображення і міру стискування. Відмітимо, що для різних компонентів зображення можуть бути свої МК.

Наступний крок алгоритму полягає в перетворенні отриманої матриці 8x8 у вектор з 64 елементів. Цей етап називається зигзаг-скануванням, оскільки елементи з матриці вибираються, починаючи з лівого верхнього, зигзагом по діагоналях, паралельних побічній діагоналі. В результаті виходить вектор, в початкових позиціях якого знаходяться елементи матриці, відповідні низьким частотам, а в кінцевих - високим. Отже, в кінці вектора буде дуже багато нульових елементів.

Далі повторюються всі дії, відповідні стандартному алгоритму стискування нерухомих зображень JPEG.

Використання векторів зсувів блоків

Простим способом обліку подібності сусідніх кадрів було б віднімання кожного блоку поточного кадру з кожного блоку попереднього. Проте набагато ефективнішим є алгоритм пошуку векторів, на які зрушилися блоки поточного кадру по відношенню до попереднього.

Алгоритм полягає в тому, що для кожного блоку зображення ми знаходимо блок, близький до йому в деякій метриці (наприклад, по мінімуму суми квадратів різниць пікселів), в попередньому кадрі в деякій околиці поточного положення блоку. Якщо мінімальна відстань між блоками в цій метриці менше деякого порогу, то разом з кожним блоком у вихідному потоці зберігається вектор зсуву - координати зсуву максимально схожого блоку в попередньому I або P-кадрі. Якщо відмінності більше цього порогу, блок стискується незалежно.

2.10. Методи стискування мовних сигналів

Що б не говорили, а основним способом спілкування і зв'язку між людьми були і залишаються мова і передача мовних повідомлень. Основні об'єми переданої в системах зв'язку інформації сьогодні доводиться на мову – це і дротяна телефонія, і системи стільникового і супутникового зв'язку, і так далі Тому

ефективному кодуванню, або стискуванню мови, в системах зв'язку приділяється виняткова увага.

Історія стискування мовних сигналів в системах зв'язку налічує вже не один десяток років. Так, наприклад, в ті часи, коли час чекання замовленої телефонної розмови складав десятки годинника, економічні обмеження привели до установки на трансконтинентальних лініях США і атлантичному кабелі так званої апаратури J2, канали якої мали смугу 0,3 - 1,7 кГц при необхідній для нормальної якості зв'язку смузі 0,3 – 3,5 кГц. Така апаратура колись працювала і на лінії Москва -Владивосток. Якість її каналів ледве досягала двох балів MOS, але вирішальним виявилося двократне збільшення числа телефонних з'єднань. Потреби користувачів в каналах зробили тоді питання якості мови другорядними. Сьогодні ж чинник якості є не менш важливим, чим економія пропускної спроможності каналів зв'язку.

Розглянемо основні властивості мовного сигналу як об'єкту економного кодування і передачі по каналах зв'язку і спробуємо пояснити, на яких властивостях сигналу грунтується можливість його стискування.

Мова є коливаннями складної форми, залежної від вимовних слів, тембру голосу, інтонації, підлоги і віку того, що говорить. Спектр мови вельми широкий (приблизно від 50 до 10000 Гц), але для передачі мови в аналоговій телефонії колись відмовилися від складових, лежачих за межами смуги 0,3 - 3,4 кГц, що дещо погіршило сприйняття ряду звуків (наприклад шиплячих, істотна частина енергії яких зосереджена у верхній частині мовного спектру), але мало торкнулося розбірливості. Обмеження частоти знизу (до 300 Гц) також трохи погіршує сприйняття із-за втрат низькочастотних гармонік основного тону.

На приведених нижче малюнках змальовані фрагменти мовних сигналів, що містять явні (рис.2.16) і приголосні (рис.2.17) звуки, а також спектри цих сигналів (рис.2.18 і 2.19). Добре видні різниця в характері відповідних сигналів, а також те, що як в першому, так і в другому випадках ширина спектру сигналу не перевищує 3,5 кГц. Окрім цього, можна відзначити, що рівень низькочастотних (тобто повільних за часом) складових в спектрі мовного сигналу значно вищий за рівень високочастотних (швидких) складових. Ета істотна нерівномірність спектру, до речі, є одним з чинників стисливості таких сигналів.

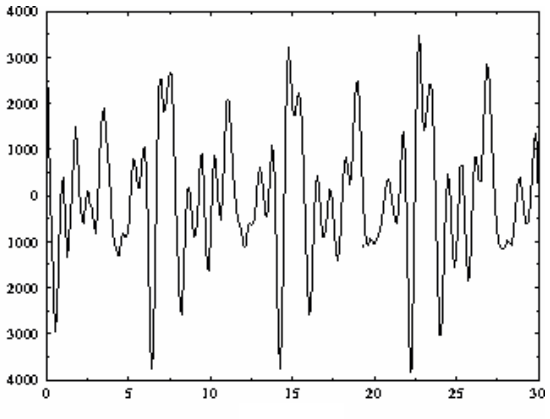


Рис. 2.16

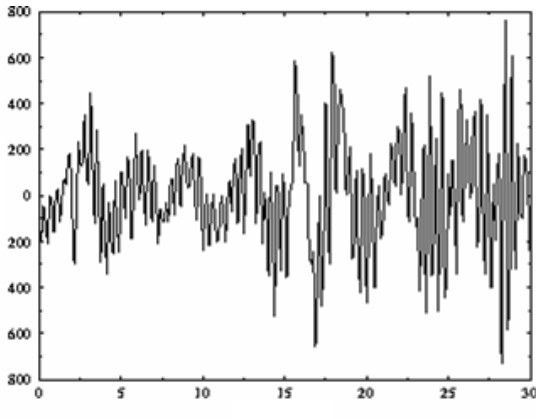


Рис. 2.17

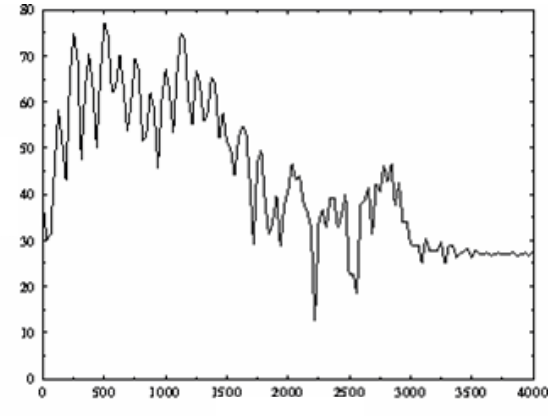


Рис. 2.18

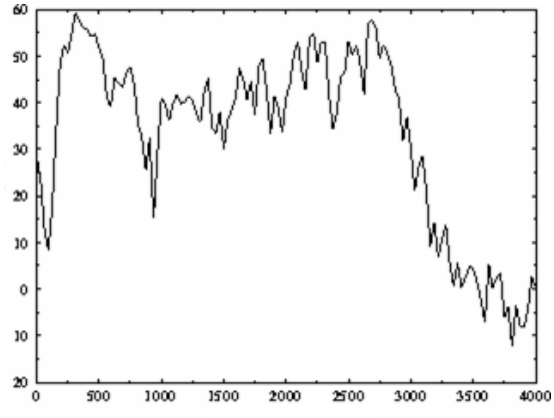


Рис. 2.19

Другою особливістю мовних сигналів, як це можна відзначити з наведених прикладів, є нерівномірність розподілу вірогідності (щільність вірогідності) миттєвих значень сигналу. Малі рівні сигналу значно вірогідніші, ніж великі. Особливо це помітно на фрагментах великої тривалості з невисокою активністю мови. Цей чинник, як відомо, також забезпечує можливість економного кодування – вірогідніші значення можуть кодуватися короткими кодами, менш вірогідні – довгими.

Ще одна особливість мовних сигналів – їх істотна нестаціонарна в часі: властивості і параметри сигналу на різних ділянках значно розрізняються. При цьому розмір інтервалу стаціонарності складає порядка декілька десятків мілісекунд. Це властивість сигналу значно утрудняє його економне кодування і заставляє робити системи стискування адаптивними, тобто що підстроюються під значення параметрів сигналу на кожній з ділянок.

Нарешті, виключно важливим для організації стискування мовних сигналів є розуміння фізики механізму речеобразования. Його спрощена схема приведена на рис. 2.20.

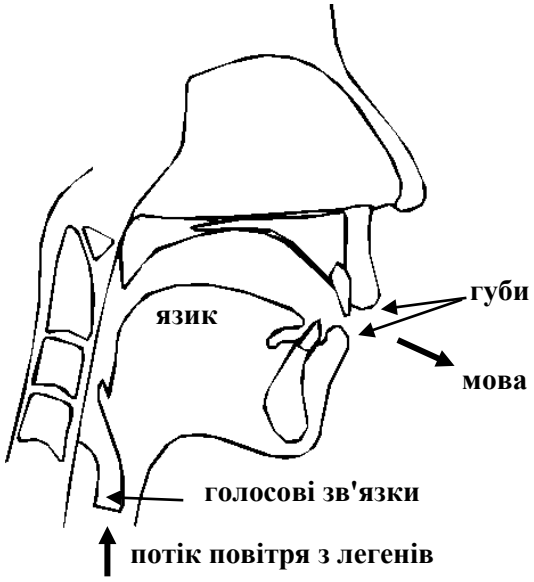


Рис. 2.20

Мова формується при проходженні виштовхуваного легенями потоку повітря через голосові в'язки і голосовий тракт. Голосовий тракт починається від голосових в'язок і закінчується губами і в середньому має довжину порядка 15 - 17 сантиметрів. Голосовий тракт через свої резонансні властивості вносить до формованого сигналу набір характерних для кожної людини частотних складових, званих формантами. Частоти і смуги цих формант можуть управлятися зміною форми голосового тракту, наприклад, зміною положення мови. Важливою частиною багато голосових кодеров/декодеров є моделювання голосового тракту як короткочасного фільтру із змінними параметрами. Оскільки форма голосового тракту може змінюватися порівняно повільно (важко передбачити, що можна змінювати положення мови частіше, ніж 20 – 30 разів в секунду), то параметри такого фільтру повинні оновлюватися (або змінюватися) також порівняно рідко (зазвичай – через кожних 20 мілісекунд або навіть рідше).

Таким чином, голосовий тракт збуджується потоком повітря, що направляється в нього через голосові в'язки. Залежно від способу збудження звуки, що виникають при цьому, можна розділити на три класи:

1. Явні звуки, що виникають, коли голосові в'язки вібрують, відкриваючись і закриваючись, перериваючи тим самим потік повітря від легенів до голосового тракту. Збудження голосового тракту при цьому виробляється квазіперіодичними імпульсами. Швидкість (частота) відкриття і закривання в'язок визначають висоту виникаючого звуку (тони). Вона може управлятися зміною форми і напруги голосових в'язок, а також зміною тиску повітряного потоку, що підводиться.

Явні звуки мають високу міру періодичності основного тону з періодом 2 - 20 мс. Ета довготривала періодичність добре видно на рис.2.17, де приведений фрагмент мовного сигналу з явним звуком.

2. Приголосні звуки, що виникають при збудженні голосового тракту шумоподобным турбулентним потоком, формованим проходящим з високою швидкістю через відкриті голосові в'язки потоком повітря. У таких звуках, як це видно з рис.2.17, практично відсутня довготривала періодичність, обумовлена вібрацією голосових в'язок, проте короткочасна кореляція, обумовлена впливом голосового тракту, має місце.

3. Звуки вибухового характеру, що виникають, коли закритий голосовий тракт з надлишковим тиском повітря раптово відкривається.

Деякі звуки в чистому вигляді не личать ні під один з описаних вище класів, але можуть розглядатися як їх суміш.

Таким чином, процес мовоутворення можна розглядати як фільтрацію мовноутворюючим трактом з параметрами сигналів збудження, що змінюються в часі, також з характеристиками, що змінюються (рис. 2.21).

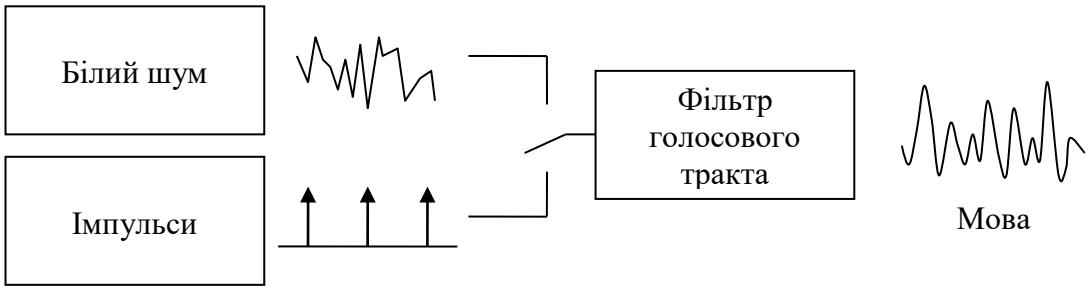


Рис. 2.21

При цьому, не дивлячись на виняткову різноманітність мовних сигналів, що генеруються, форма і параметри голосового тракту, а також способи і параметри збудження досить одноманітні і змінюються порівняно повільно. З рис.2.16 і 2.17 добре видно, що мовний сигнал володіє високою мірою короткочасної і довготривалої передбаченості із-за періодичності вібрацій голосових в'язок і резонансних властивостей голосового тракту. Більшість кодеров/декодеров мови і використовують цю передбаченість, а також повільність зміни параметрів моделі системи речеобразования для зменшення швидкості коди.

Література

1. Лезин Ю.С. Введение в теорию и технику радиотехнических систем. - М.: Радио и связь, 1986.
2. Зюко А.Г. Теория передачи сигналов. - М.: Сов. радио, 1972.
3. Радиотехнические системы /Под ред. Ю.М. Казаринова - М.: Сов. радио, 1968.
4. Чердынцев В.А. Радиотехнические системы. – Минск: Вышэйш. шк., 1988.
6. Пенин П.И. Системы передачи цифровой информации. - М.: Сов. радио, 1976.
7. Мордухович Л.Г., Степанов А.Г. Системы радиосвязи (курсовое проектирование). - М.: Радио и связь, l987.
8. Системы радиосвязи /Под ред. Н.К. Калашникова - М.: Радио и связь, 1988.
9. Пышкин И. Н. Системы первичной радиосвязи. - М.: Радио и связь, 1988.
15. Тепляков И.П., Рощин Б.В. Радиосистемы передачи информации. - М.: Радио и связь, 1982.
16. Банкет В.Л., Дорофеев В.П. Цифровые методы в спутниковой связи. - М.: Радио и связь, 1988.
18. Кузьмин И.В. Основы теории информации и кодирования. – Минск: Вышэйш. шк., 1986.
20. Хемминг Р.В. Теория информации и теория кодирования. - М.: Радио и связь, 1983.
21. Ватолин Д.С. Алгоритмы сжатия изображений. – М: МГУ, 1999. http://graphics.cs.msu.su/library/our_publications/index.htm.
22. Ватолин Д.С. MPEG-стандарт ISO на видео в системах мультимедиа. E-mail: rois@red.com.ru.
23. Coding of moving pictures and associated audio. Committee Draft of Standart ISO11172: ISO/MPEG 90/176 Dec., 1990.
24. Le Gall D.A. MPEG: a video compression standart for multimedia applications. Communication of ACM. Volume 34. Number 4., April 1991.
25. JPEG digital compression and coding of continuus-tone still images. Draft ISO 10918, 1991.