Formalising Mathematics - Project 2

Francesco N. Chotuck

Contents

1 Overview of the project

The aim of this project is to formalise a personal favourite result from the second year undergraduate curriculum: the Banach fixed-point theorem, also known as the contraction mapping theorem.

This work establishes several key results. It begins by defining a contraction mapping on a metric space, followed by the derivation of essential inequalities concerning a particular 'contracting' sequence, which will be instrumental in the proof. The project culminates in the formalisation of the existence and uniqueness of a fixed point in a complete metric space, ultimately leading to the formalisation of the Banach fixed-point theorem.

Banach originally established what is now known as the "Banach fixed-point theorem" in [Ban22], within the context of what are now termed *Banach spaces*. Subsequently, Renato Caccioppoli independently generalised the result to complete metric spaces [Cac23]. In this work, we follow the proof presented in [Lee03, Appendix C] in the setting of complete metric spaces.

For a comprehensive historical account of this theorem and further results by Banach, see [JJT24].

1.1 Sketch of the proof

Before presenting the Lean formalisation, we provide a brief sketch of the proof, as the reasoning behind the theorem may not be immediately clear from reading blocks of code and seemingly arbitrary lemma statements.

The Banach fixed-point theorem states that a contraction mapping T on a complete metric space X has a unique fixed point. The proof follows the steps below.

1. **Definition of the iterative sequence:** Begin with an arbitrary initial point x_0 and construct the sequence $\{x_n\}_{n=0}^{\infty}$ by iterating T, i.e., $x_{n+1} = T(x_n)$.

2. **Demonstrating that the sequence is Cauchy:** The contraction property,

$$d(T(x), T(y)) \leq q d(x, y), \quad \text{for some } q \in [0, 1),$$

   ensures that the distances between successive terms decrease geometrically:

$$d(x_{n+1}, x_n) \leq q^n d(x_1, x_0).$$

   By summing over n, it follows that $\{x_n\}_{n=0}^{\infty}$ is a Cauchy sequence.

3. **Convergence to a fixed point:** Since X is a complete metric space, the Cauchy sequence $\{x_n\}_{n=0}^{\infty}$ converges to some limit x'. The continuity of T then implies that $T(x') = x'$, establishing the existence of a fixed point.

4. **Uniqueness of the fixed point:** Suppose x' and y' are two fixed points of T. By applying the contraction property, we obtain

$$d(x', y') = d(T(x'), T(y')) \leq q d(x', y').$$

   Since $q \in [0, 1)$, this forces $d(x', y') = 0$, and thus $x' = y'$, proving uniqueness.

To ultimately prove that the sequence is Cauchy, we must formalise several useful inequalities, as illustrated in Section 2.2.

2 The code

This section presents the complete code developed for the project. Each result is first introduced with its informal statement and proof, then followed by its formalisation in Lean.

The code starts by importing the required Mathlib API:

```
import Mathlib.Algebra.EuclideanDomain.Basic
import Mathlib.Algebra.EuclideanDomain.Field
import Mathlib.Analysis.Normed.Order.Lattice
import Mathlib.Analysis.SpecialFunctions.Log.Basic
import Mathlib.Data.Real.StarOrdered
import Mathlib.Topology.EMetricSpace.Paracompact
```

Initially, this is done using `import Mathlib`. Once the implementation of the project is complete, the command `#min_imports` is executed at the end of the code to ensure that only the essential dependencies are retained. This technique was introduced to me by Yuming Guo, who, in turn, learned it from Sidharth Hariharan.

2.1 Definitions

We start by defining the notion of a contraction.

Definition 2.1 ([Lee03, Page 657])**.** Let X be a metric space. A map $T : X \to X$ is said to be a **contraction** if there is a constant $q \in [0, 1)$ such that $d(T(x), T(y)) \leq qd(x, y)$ for all $x, y \in X$.

Remark 2.2. In [Lee03], the contraction constant is defined within the interval $q \in (0, 1)$. However, in my first course on "Metric Spaces and Topology", we also permitted $q = 0$, for which the theorem remains valid. Some sources, as referenced in [Rho77], adopt the convention $q \in [0, 1)$. Accordingly, for the purposes of this project, we modify Lee's definition to include $q = 0$.

In Lean we formalise this definition as follows.

```
def is_contraction {X : Type} [MetricSpace X] (T : X → X) (q : ℝ) : Prop :=
    0 ≤ q ∧ q < 1 ∧ ∀ x y : X, dist (T x) (T y) ≤ q * dist x y
```

We could have incorporated the conditions on the contraction constant q directly into the hypothesis of the definition. However, we deemed it more appropriate not to do so, as this approach facilitates the extraction of conditions on q in subsequent lemmas and theorems.

Next, we define the concept of a fixed point, which is central to this project. However, formalising this notion separately is unnecessary, as it can be incorporated into the hypothesis of the main theorem in the form of `T x = x`, as will be demonstrated later.

Definition 2.3 ([Lee03, Page 657])**.** A **fixed point** of a map $F : X \to X$ is a point $x \in X$ such that $F(x) = x$.

As outlined in Section 1.1 we must define the sequence $x_n = T(x_{n-1})$ in Lean.

```
def seq {X : Type} (T : X → X) (x₀ : X) : ℕ → X
    | 0     => x₀
    | (n + 1) => T (seq T x₀ n)
```

In the Lean definition of this sequence, it is unnecessary to specify that T is a contraction or that X is a metric space. This is because these properties are not required for the mere definition of the sequence; their relevance arises only in the proof of the theorem. Moreover, including such assumptions prematurely would result in unused arguments, which Lean's `#lint` tool would flag.

2.2 Summation inequalities

As outlined in Section 1.1, we must demonstrate that the sequence $\{x_n\}_{n=0}^{\infty}$ is Cauchy. To achieve this, we first establish a set of useful inequalities, which will be formulated as lemmas.

Unless stated otherwise, we will now focus solely on the sequence $x_n = T(x_{n-1})$. In the following lemmas, we implicitly refer to this sequence.

Lemma 2.4. For all $n \in \mathbb{N}$, the inequality $d(x_{n+1}, x_n) \leq q^n d(x_1, x_0)$ holds.

Informal proof. We proceed by induction on n.

For the base case $n = 0$, we have $d(x_1, x_0) \leq d(x_1, x_0)$, which is trivially true.

For the inductive step, assume the inequality holds for $n = k$. We aim to show that it holds for $n = k+1$. By the definition of the sequence, we have $d(x_{k+2}, x_{k+1}) = d(T(x_{k+1}), T(x_k))$. Since T is a contraction, it satisfies $d(T(x_{k+1}), T(x_k)) \leq q d(x_{k+1}, x_k)$.

Applying the induction hypothesis, $d(x_{k+1}, x_k) \leq q^k d(x_1, x_0)$, we obtain

$$d(x_{k+2}, x_{k+1}) \leq q(q^k d(x_1, x_0)) = q^{k+1} d(x_1, x_0),$$

as required. □

We implement this in Lean, which, despite the intricate notation, is straightforward.

```
lemma contraction_inequality (hT : is_contraction T q) (x₀ : X) :
    ∀ n : ℕ, dist (seq T x₀ (n + 1)) (seq T x₀ n) ≤ q ^ n * dist (seq T x₀ 1) x₀ := by
  intro n
  induction' n with k hk -- Perform induction on `n`
  · -- Base case: when `n = 0`, the inequality holds trivially
    simp only [seq, pow_zero, one_mul, le_refl]
  · -- Inductive case
    simp only [seq]
    -- Extract properties of the contraction mapping
    obtain ⟨h_q_nonneg, h_q_lt1, h_contraction⟩ := hT
    calc
      -- Expressing the distance between consecutive iterates
      dist (seq T x₀ (k + 2)) (seq T x₀ (k + 1))
      = dist (T (seq T x₀ (k + 1))) (T (seq T x₀ k)) := by rfl -- Definition of the sequence
      -- Apply the contraction property: `d(T(x), T(y)) ≤ q * d(x, y)`
      _ ≤ q * dist (seq T x₀ (k + 1)) (seq T x₀ k) := h_contraction (seq T x₀ (k + 1)) (seq T x₀ k)
      -- Use the induction hypothesis: `d(x_{k+1}, x_k) ≤ q^k * d(x₁, x₀)`
      _ ≤ q * (q ^ k * dist (seq T x₀ 1) x₀) := by exact mul_le_mul_of_nonneg_left hk h_q_nonneg
      -- Rewriting to group terms correctly
      _ ≤ (q * q ^ k) * dist (seq T x₀ 1) x₀ := by rw [mul_assoc]
      -- Using exponentiation property: `q * q^k = q^(k+1)`
      _ = q ^ (k + 1) * dist (seq T x₀ 1) x₀ := by rw [pow_succ']
```

The next set of lemmas will be in order to prove the following inequalities,

$$\begin{aligned} d(x_m, x_n) &\leq d(x_m, x_{m-1}) + d(x_{m-1}, x_{m-2}) + \cdots + d(x_{n+1}, x_n) \\ &\leq q^{m-1} d(x_1, x_0) + q^{m-2} d(x_1, x_0) + \cdots + q^n d(x_1, x_0) \\ &= q^n d(x_1, x_0) \sum_{k=0}^{m-n-1} q^k \\ &\leq q^n d(x_1, x_0) \sum_{k=0}^{\infty} q^k \\ &= q^n d(x_1, x_0) \left(\frac{1}{1-q} \right). \end{aligned}$$

We sketch a proof of the algebraic manipulation outlined above. By the triangle inequality, the distance $d(x_m, x_n)$ is bounded by the sum of successive distances along the sequence. Applying the bound $d(x_{k+1}, x_k) \leq q^k d(x_1, x_0)$ from Lemma 2.4, we express the sum in terms of a geometric series. Taking the infinite geometric sum of the upper bound yields the result.

For the Lean implementation, each inequality is proven in a separate lemma to maintain clarity and avoid an overly cumbersome `calc` block.

Lemma 2.5. For any sequence $\{x_n\}_{n=0}^{\infty}$ the inequality $d(x_m, x_n) \leq \sum_{k=n}^{m-1} d(x_{k+1}, x_k)$.

Remark 2.6. It is important to emphasise that this inequality holds for any sequence, a fact that will be crucial in the Lean implementation of the inequalities above.

We reformulated the inequality using summation notation to facilitate its implementation in Lean.

```
lemma dist_seq_telescoping (x : ℕ → X) (m n : ℕ) (h : n ≤ m) :
    dist (x m) (x n)
      ≤ Finset.sum (range (m - n)) (fun k ↦ dist (x (n + k)) (x (n + k + 1))) := by
  -- Express ‘m‘ as ‘n + d‘, where ‘d = m - n‘
  obtain ⟨d, rfl⟩ := Nat.exists_eq_add_of_le h
  induction' d with d hd -- Proceed by induction on ‘d‘
  · -- Base case: when ‘d = 0‘, we have ‘m = n‘, so ‘dist(x n, x n) = 0‘
    simp only [add_zero, dist_self, tsub_self, range_zero, sum_empty, le_refl]
  · -- Inductive step: Assume the result holds for ‘d‘, prove it for ‘d + 1‘
    simp only [add_tsub_cancel_left]
    -- Inductive hypothesis: The sum of distances up to ‘d‘ satisfies the inequality
    have ih' : dist (x (n + d)) (x n)
        ≤ Finset.sum (range d) (fun k ↦ dist (x (n + k)) (x (n + k + 1))) := by aesop
    calc
      -- Apply the triangle inequality to include ‘x(n + d)‘
      dist (x (n + d.succ)) (x n)
          ≤ dist (x (n + d.succ)) (x (n + d)) + dist (x (n + d)) (x n) :=
            dist_triangle (x (n + d.succ)) (x (n + d)) (x n)
      -- Use the induction hypothesis to bound ‘dist(x(n + d), x n)‘
      _ ≤ dist (x (n + d.succ)) (x (n + d))
          + Finset.sum (range d) (fun k ↦ dist (x (n + k)) (x (n + k + 1))) :=
            add_le_add (le_refl (dist (x (n + d.succ)) (x (n + d)))) ih'
      -- Rewriting to express the sum in terms of ‘range d.succ‘
      _ = Finset.sum (range d.succ) (fun k ↦ dist (x (n + k)) (x (n + k + 1))) := by
        rw [add_comm, sum_range_succ, ← dist_comm (x (n + d)) (x (n + d.succ))]
        rfl
```

Remark 2.7. Despite using `open Finset`, the syntax `Finset.sum` remains protected, preventing the use of `sum` alone. In contrast, for `Finset.range`, the prefix `Finset` can be omitted.

In the formalisation, we express m as $n + d$, where $d = m - n$, and proceed by induction on d. This approach allows us to utilise theorems and lemmas from the Mathlib API whose indices start from 0. The base case $d = 0$ is trivial as both sides are zero. For the inductive step, applying the triangle inequality gives

$$d(x_{n+d+1}, x_n) \leq d(x_{n+d+1}, x_{n+d}) + d(x_{n+d}, x_n).$$

Using the induction hypothesis, the second term is bounded by a sum over $d - 1$ terms, leading to

$$d(x_{n+d+1}, x_n) \leq \sum_{k=0}^{d} d(x_{n+k}, x_{n+k+1}),$$

which matches the desired bound.

In Lean, it was nececssary to shift the index of the summation to ensure that the recursive structure aligns with the desired inequality. The key step involves rewriting the sum to include the next term, achieved using `Finset.sum.range_succ`, which extends a summation by appending the final term.

The subsequent inequalities were more straightforward to implement using `calc` and search tactics.

Lemma 2.8. The inequality $d(x_m, x_n) \leq q^n d(x_1, x_0) \sum_{k=0}^{m-n-1} q^k$ holds.

```
lemma seq_dist_geometric_sum_bound (hT : is_contraction T q) (x₀ : X) (m n : ℕ) (hmn : n ≤ m) :
    dist (seq T x₀ m) (seq T x₀ n)
      ≤ q ^ n * dist (seq T x₀ 1) x₀ * Σ i in range (m - n), q ^ i := by
  -- Step 1: Establish an upper bound on the distance between `seq T x₀ m` and `seq T x₀ n`
  have hBound : dist (seq T x₀ m) (seq T x₀ n)
    ≤ Σ i in range (m - n), q ^ (n + i) * dist (seq T x₀ 1) x₀ := by
    -- Apply the telescoping distance inequality
    apply (dist_seq_telescoping (seq T x₀) m n hmn).trans
    -- Show that summing over distances preserves the inequality
    apply sum_le_sum
    intro i _
    -- Rewrite the distance using symmetry
    rw [dist_comm]
    -- Apply the contraction inequality to bound each term
    exact contraction_inequality hT x₀ (n + i)
  -- Apply the contraction inequality to bound each term
  calc
    -- Use the previously established bound
    dist (seq T x₀ m) (seq T x₀ n)
      ≤ Σ i in range (m - n), q ^ (n + i) * dist (seq T x₀ 1) x₀ := hBound
    -- Rewrite the exponentiation using the property `q^(n + i) = q ^ n * q^i`
    _ = Σ i in range (m - n), (q ^ n * q ^ i) * dist (seq T x₀ 1) x₀ := by
          simp only [pow_add, mul_assoc]
    -- Factor out `q ^ n * dist (seq T x₀ 1) x₀` from each term in the sum
    _ = Σ i in range (m - n), (q ^ n * dist (seq T x₀ 1) x₀) * q ^ i := by
          apply sum_congr rfl
          intro i hi
          nth_rewrite 4 [mul_comm] -- Reorder multiplication
          rw [mul_comm, mul_assoc]
    -- Factor `q ^ n * dist (seq T x₀ 1) x₀` out of the entire sum
    _ = q ^ n * dist (seq T x₀ 1) x₀ * Σ i in range (m - n), q ^ i :=
        by rw [mul_sum]
```

We note that Lemma 2.5 was applied to the sum under consideration, as it holds for any sequence, including the iterative contraction sequence used for this project.

Lemma 2.9. The inequality $\sum_{k=0}^{m-n-1} q^k \leq \sum_{k=0}^{\infty} q^k$ holds.

```
lemma partial_sum_le_infinite_sum (hq0 : 0 ≤ q) (hq1 : q < 1) (m n : ℕ) :
    Σ k in range (m - n), q ^ k ≤ Σ' k, q ^ k := by
  by_cases h : n ≤ m
  -- Case 1: `n ≤ m`, so `(m - n)` is nonnegative
  · have h_nonneg : ∀ k, 0 ≤ q ^ k := fun k ↦ pow_nonneg hq0 k
    -- Apply the standard inequality: finite sum is bounded by the infinite sum
    refine sum_le_tsum (range (m - n)) (fun i a ↦ h_nonneg i) ?_
    -- Shows that the geometric series `Σ' q^k` is summable
    exact summable_geometric_of_lt_one hq0 hq1
  · -- Case 2: `n > m`, so `(m - n)` is negative (empty sum)
    refine sum_le_tsum (range (m - n)) ?_ ?_
    -- Shows that each term in the finite sum is nonnegative
    exact fun i a ↦ pow_nonneg hq0 i
    -- Shows that the infinite geometric series is summable
    exact summable_geometric_of_lt_one hq0 hq1
```

Implementing this lemma in Lean requires accounting for the case where $n > m$, a scenario typically overlooked in informal proofs. It is surprising to observe that Lean requires this consideration. In this case, the sum is naturally empty and is therefore defined as 0. This is managed using the `by_cases` tactic.

Lemma 2.10. The inequality $d(x_m, x_n) \leq q^n d(x_1, x_0) \left(\frac{1}{1-q}\right)$ holds.

```
lemma seq_dist_bound_simplified (hT : is_contraction T q) (x₀ : X) (m n : ℕ) (hmn : n ≤ m) :
    dist (seq T x₀ m) (seq T x₀ n) ≤ q ^ n * dist (seq T x₀ 1) x₀ * (1 - q)⁻¹ := by
  -- Step 1: Use the previously established bound on the sequence distance
  have h1 := seq_dist_geometric_sum_bound hT x₀ m n hmn
  -- Step 2: The partial sum of the geometric series is bounded above by the infinite sum
  have h2 : Σ i in range (m - n), q ^ i ≤ Σ' i, q ^ i :=
    partial_sum_le_infinite_sum hT.1 hT.2.1 m n
  -- Step 3: The closed-form sum of the infinite geometric series
  have h3 : Σ' i, q ^ i = (1 - q)⁻¹ :=
    tsum_geometric_of_lt_one hT.1 hT.2.1
  -- Step 4: Applying the inequalities to obtain the final bound
  calc
    dist (seq T x₀ m) (seq T x₀ n)
        ≤ q ^ n * dist (seq T x₀ 1) x₀ * Σ i in range (m - n), q ^ i := h1
    -- Use the upper bound on the partial sum
    _ ≤ q ^ n * dist (seq T x₀ 1) x₀ * Σ' i, q ^ i := by
      apply mul_le_mul_of_nonneg_left h2
      -- Show that 'q ^ n * d(x₁, x₀)' is nonnegative
      apply mul_nonneg
      · exact pow_nonneg hT.1 n
      · exact dist_nonneg
    _ = q ^ n * dist (seq T x₀ 1) x₀ * (1 - q)⁻¹ := by rw [h3]
```

Possibly, this chain of inequalities could have been proven in a single lemma using a large `calc` block. However, we chose to break down each inequality into separate lemmas for stylistic and computational reasons.

The next result establishes that the expression $q^n d(x_1, x_0) \left(\frac{1}{1-q}\right)$ can be rearranged into $q^n < \frac{\varepsilon(1-q)}{d(x_1,x_0)}$.

```
lemma rearrangement_seq_final (hT : is_contraction T q) (x₀ : X) (n : ℕ) (ε : ℝ)
    (h : q ^ n * dist (seq T x₀ 1) x₀ * (1 - q)⁻¹ < ε) :
    (dist (seq T x₀ 1) x₀ = 0 ∨ q ^ n < ε * (1 - q) * (dist (seq T x₀ 1) x₀)⁻¹) := by
  by_cases hd : x₀ = seq T x₀ 1
  · left
    simp only [dist_eq_zero]
    exact (Eq.symm hd)
  · right
    have h_d_pos : 0 < dist (seq T x₀ 1) x₀ := dist_pos.mpr fun a ↦ hd ((Eq.symm a))
    have h_1q_pos : 0 < 1 - q := one_minus_q_pos hT
    rw [propext (lt_mul_inv_iff₀ h_d_pos)]
    nth_rewrite 2 [mul_comm]
    rw [← propext (mul_inv_lt_iff₀' h_1q_pos)]
    exact h
```

Although this result might appear redundant and could have been proven as a `have` statement within the formalisation of the theorem where this rearrangement is applied, doing so proved to be unexpectedly challenging. Therefore, it was more practical to formalise this result as a separate `lemma`.

Once again, it is necessary to consider two cases: in the trivial scenario where $x_1 = x_0$, the rearrangement leads to $\varepsilon > 0$. Otherwise, the desired result follows. This case analysis is managed using the `by_cases` tactic.

The remaining results are straightforward consequences and are therefore presented

without detailed proofs. The lemma `one_minus_q_pos` is stated separately because it is frequently used throughout the formalisation, making it more convenient to reference. Additionally, the lemma `ceil_add_one_le_nat_imp_lt` establishes that for $a \in \mathbb{R}$ and $n \in \mathbb{N}$, the inequality $\lceil a \rceil + 1 \leq n$ implies $a < n$. This result is utilised in the formalisation of Theorem 2.11.

2.3 The sequence is Cauchy

With the necessary inequalities formalised, the next step is to formalise a crucial result required for the implementation of the proof of the Banach Fixed-Point Theorem.

Theorem 2.11. The sequence $\{x_n\}_{n=0}^{\infty}$ is Cauchy.

Informal proof. This proof follows a standard approach. The objective is to show that for every $\varepsilon > 0$, there exists an N such that for all $m, n \geq N$, $d(x_m, x_n) < \varepsilon$. From the previously established inequalities, it follows that

$$d(x_m, x_n) \leq q^n d(x_1, x_0) \left(\frac{1}{1-q} \right).$$

Since $q < 1$, it follows that $q^n \to 0$ as $n \to \infty$. Therefore, it is possible to select a sufficiently large N such that

$$q^N < \frac{\varepsilon(1-q)}{d(x_1, x_0)}.$$

By rearranging this inequality, we choose

$$N = \left\lceil \frac{\log\left(\frac{\varepsilon(1-q)}{d(x_1,x_0)}\right)}{\log q} \right\rceil + 1$$

where the ceiling function is used to ensure that N is a natural number. This choice of N guarantees that for all $m, n \geq N$,

$$d(x_m, x_n) \leq q^n d(x_1, x_0) \left(\frac{1}{1-q} \right) < \varepsilon,$$

as required. □

Remark 2.12. This choice of N was made to enable the application of the lemma `ceil_add_one_le_nat_imp_lt` in the subsequent series of rewrites within the formalisation. Although more elegant rewrites might exist, time constraints necessitated this approach.

The formalisation of this theorem proved to be the most challenging aspect of the project. Incorporating the straightforward algebraic manipulations outlined in the previous section was unexpectedly difficult to implement in Lean. This difficulty was likely due to inexperience in effectively searching Mathlib for the relevant lemmas needed to facilitate these calculations.

```
theorem seq_is_cauchy (hT : is_contraction T q) (x₀ : X) (hq : 0 < q) : CauchySeq (seq T x₀) := by
  -- Rewrite the Cauchy sequence definition in metric spaces
  rw [Metric.cauchySeq_iff]
  intro ε hε
  -- ‘q < 1‘ from the contraction property
  have q_lt1 := hT.2.1
  have h_one_minus_q_pos : 0 < 1 - q := one_minus_q_pos hT
  -- Define ‘N‘ such that the geometric term is sufficiently small
  set N := ⌈(Real.log ((ε * (1 - q)) / dist (seq T x₀ 1) x₀)) / Real.log q⌉₊ + 1 with hN
  use N
  intro m hm n hn
  -- Without loss of generality, assume ‘n ≤ m‘
  wlog hmn : n ≤ m generalizing m n
  · rw [dist_comm]
    exact this n hn m hm (le_of_not_le hmn)
  · -- Case 1: The sequence stabilizes if ‘x₀ = seq T x₀ 1‘
    by_cases h_terms_eq : x₀ = seq T x₀ 1
    · have h_dist_eq0 : dist (seq T x₀ 1) x₀ = 0 := by
        simp only [dist_eq_zero]
        exact (Eq.symm h_terms_eq)
      calc
        _ ≤ q ^ n * dist (seq T x₀ 1) x₀ * (1 - q)⁻¹ := seq_dist_bound_simplified hT x₀ m n hmn
        _ ≤ q ^ n * 0 * (1 - q)⁻¹ := by
          exact le_of_eq (congrFun (congrArg HMul.hMul
          (congrArg (HMul.hMul (q ^ n)) h_dist_eq0)) (1 - q)⁻¹)
        _ = 0 := by field_simp
        _ < ε := by exact hε
    · -- Case 2: Apply the geometric bound for contractions
      have h_d_pos : dist (seq T x₀ 1) x₀ > 0 := by
        exact dist_pos.mpr fun a ↦ h_terms_eq (Eq.symm a)
      have hN' : q ^ n * dist (seq T x₀ 1) x₀ * (1 - q)⁻¹ < ε := by
        rw [hN] at hn
        simp only [ge_iff_le] at hn
        apply ceil_add_one_le_nat_imp_lt at hn
        rw [div_lt_iff_of_neg] at hn
        · rw [← Real.log_pow, Real.log_lt_iff_lt_exp] at hn
          · rw [Real.exp_log] at hn
            · rw [lt_div_iff₀] at hn
              · rw [← inv_mul_lt_iff₀'] at hn
                · rw [← mul_assoc] at hn
                  rw [mul_comm, ← mul_assoc]
                  exact hn
                · exact h_one_minus_q_pos
              · simp only [dist_pos, ne_eq]
                exact fun a ↦ h_terms_eq ((Eq.symm a))
            · positivity
          · positivity
        · exact Real.log_neg hq q_lt1
      calc
        dist (seq T x₀ m) (seq T x₀ n)
          ≤ q ^ n * dist (seq T x₀ 1) x₀ * (1 - q)⁻¹ := by
            exact seq_dist_bound_simplified hT x₀ m n hmn
          _ = q ^ n * (dist (seq T x₀ 1) x₀ * (1 - q)⁻¹) := by field_simp
          _ < ε * (1 - q) * (dist (seq T x₀ 1) x₀)⁻¹ * (dist (seq T x₀ 1) x₀ * (1 - q)⁻¹) := by
            gcongr
            have := rearrangement_seq_final hT _ _ _ hN'
            cases' this with h_left h_right
            · rw [h_left]
              aesop
            · exact h_right
          _ = ε := by field_simp
```

Two points are worth noting. First, this formalisation applies only for $q > 0$. Although the result is trivially true for the constant sequence of zeroes that arises when $q = 0$,

this case is deliberately excluded from the current implementation. Instead, in Lemma 2.14, it is shown that the constant sequence of zeroes converges to a limit. This approach, suggested by Bhavik Mehta, simplifies the formalisation of the sequence being Cauchy.

Once again, it is necessary to consider the trivial case when $x_1 = x_0$, using similar methods as previously employed. The greatest challenge arose from the use of `rw`'s to perform simple algebraic manipulations, which resulted in awkward and unattractive indentation within the code. In hindsight, it may have been more effective to decompose this theorem into smaller, more 'atomic' lemmas to improve readability and maintainability.

It is worth noting that the statement

```
have hN' : q ^ n * dist (seq T x₀ 1) x₀ * (1 - q)⁻¹ < ε := by ...
```

could have been proven using a different approach, specifically by employing notions of convergence, using `Filters`. This would involve demonstrating that

$$\lim_{n\to\infty} q^n d(x_1, x_0)\left(\frac{1}{1-q}\right) \to 0.$$

Although this method was initially attempted, difficulties arose in fully implementing it. Specifically, while it was possible to show that the left-hand side tends to zero, the next steps were unclear. Consequently, the decision was made to submit the more complete, `sorry`-free version of the code, prioritising completeness and correctness over an incomplete alternative.

2.4 Existence and uniqueness of the fixed point

The general strategy to prove the existence of a fixed point in a complete metric space is to establish the following:

$$x' = \lim_{n\to\infty} x_n = \lim_{n\to\infty} x_{n+1} = \lim_{n\to\infty} T(x_n) = T\left(\lim_{n\to\infty} x_n\right) = T(x').$$

The detailed justification for each equality will be provided in the corresponding informal proofs.

To prove the existence of a fixed point, it is first necessary to establish that contraction mappings are continuous. This allows the interchange of the limit and the contraction operator.

Lemma 2.13 ([Sut75, Lemma 17.25])**.** A contraction mapping is continuous.

Informal proof. We aim to prove the standard ε-δ continuity criterion in a metric space, specifically:

$$\forall \varepsilon > 0,\ \exists \delta > 0 \text{ such that } d(x,y) < \delta \implies d(T(x), T(y)) < \varepsilon.$$

We will consider two cases:

1. When $q = 0$ the proof is straightforward since $d(T(x), T(y)) = 0$ for all x and y. Therefore, T is trivially continuous.

2. When $q \in (0, 1)$ we select $\delta = \frac{\varepsilon}{q}$. With this choice, we have:

$$d(T(x), T(y)) \leq qd(x, y) < q\delta,$$

which implies $d(T(x), T(y)) < \varepsilon$, as required. □

As done previously, the different cases are handled systematically in Lean.

```
lemma contraction_is_continuous (hT : is_contraction T q) :
    Continuous T := by
  rw [is_contraction] at hT
  rw [Metric.continuous_iff]
  intro b ε hε
  rcases hT with ⟨h_q_gt0, h_q_lt1, h_contraction⟩
  -- Case 1: If ‘q = 0‘, the mapping is constant and thus continuous
  by_cases h_q_eq0 : q = 0
  · use ε
    constructor
    · exact hε
    · intro a h_dist_ε_1
      specialize h_contraction a b
      rw [h_q_eq0] at h_contraction
      subst h_q_eq0
      aesop
  -- Case 2: If ‘0 < q < 1‘, use the contraction property to show continuity
  · use (ε / q)
    constructor
    · positivity
    · intro a h_dist_ε_2
      specialize h_contraction a b
      calc
      -- Apply the contraction property
        dist (T a) (T b) ≤ q * dist a b := h_contraction
        -- Scale the distance to show continuity
        _ < q * (ε / q) := by gcongr
        _ = ε := by field_simp [h_q_eq0]
```

We next demonstrate that the sequence converges to a point, denoted x', which will subsequently be shown to be the desired fixed point.

Lemma 2.14. In a complete metric space the iterative sequence $\{x_n\}_{n=0}^{\infty}$ converges to some limit, say x'.

Informal proof. This follows from the definition of a complete metric space (see [Sut75, Definition 17.2]). □

```
lemma seq_converges (hT : is_contraction T q) (x₀ : X) [CompleteSpace X] :
    ∃ x', Filter.Tendsto (seq T x₀) Filter.atTop (nhds x') := by
  -- Case 1: If ‘q = 0‘, the sequence is constant and converges to ‘T x₀‘
  obtain rfl | hq := eq_or_lt_of_le hT.1
  · use T x₀
    -- Show that the sequence is eventually constant
    have : ∀ᶠ n in Filter.atTop, seq T x₀ n = T x₀ := by
      filter_upwards [Filter.eventually_gt_atTop 0] with n hn
      cases n with
      | zero => simp at hn
      | succ n =>
          rw [seq]
          apply eq_of_dist_eq_zero
          have h_contr := hT.2.2 (seq T x₀ n) (x₀)
          rw [zero_mul] at h_contr
          apply le_antisymm
          · exact h_contr
          · exact dist_nonneg
    -- Conclude that the sequence converges to the constant limit
    have : seq T x₀ =ᶠ[Filter.atTop] fun n ↦ T x₀ := this
    apply Filter.Tendsto.congr' this.symm
    simp only [tendsto_const_nhds_iff]
  -- Case 2: If ‘0 < q < 1‘, the sequence is Cauchy
  have c : CauchySeq (seq T x₀) := seq_is_cauchy hT x₀ hq
  -- Case 2: If ‘0 < q < 1‘, the sequence is Cauchy
  exact cauchySeq_tendsto_of_complete c
```

As previously discussed in Theorem 2.11, this lemma also accounts for the case when the sequence is constant, i.e. when $q = 0$, which trivially converges to $T(x_0)$. This is accomplished using the `filter_upwards` tactic, which focuses the proof on sufficiently large indices, thereby reducing the argument to demonstrating that the sequence remains constant for all such n. The case $q > 0$ then follows trivially.

Lemma 2.15. If the sequence converges to a limit (say) x', then x' is a fixed point of the contraction mapping.

Informal proof. Since T is a contraction, it is continuous by Lemma 2.13. Applying continuity to the limit of the sequence yields

$$\lim_{n\to\infty} T(x_n) = T\left(\lim_{n\to\infty} x_n\right) = T(x').$$

By the definition of the sequence, $T(x_n) = x_{n+1}$. Hence,

$$\lim_{n\to\infty} T(x_n) = \lim_{n\to\infty} x_{n+1} = \lim_{n\to\infty} x_n = x'.$$

Since the limit of the shifted sequence x_{n+1} is identical to the limit of x_n, and because limits are unique in Hausdorff spaces (see [Sut75, Proposition 11.4]), and metric spaces are Hausdorff (see [Sut75, Proposition 11.5]), it follows that

$$T(x') = x'.$$

Therefore, x' is a fixed point of the contraction T. □

```
lemma limit_fixed (hT : is_contraction T q) (x₀ : X)
    {x' : X} (hx' : Filter.Tendsto (seq T x₀) Filter.atTop (nhds x')) : T x' = x' := by
  -- ‘T‘ is continuous
  have hc := contraction_is_continuous hT
  -- Applying continuity to the limit of the sequence
  have hlimT : Filter.Tendsto (T ∘ seq T x₀) Filter.atTop (nhds (T x')) :=
    Filter.Tendsto.comp (hc.tendsto x') hx'
  -- The sequence shifted by one step also converges to the same limit
  have hlimS : Filter.Tendsto (fun n => seq T x₀ (n + 1)) Filter.atTop (nhds x') :=
    hx'.comp (Filter.tendsto_add_atTop_nat 1)
  -- Uniqueness of limits implies that ‘T x' = x'‘
  exact tendsto_nhds_unique hlimT hlimS
```

With the prerequisite results in place, the existence of the fixed point can now be formalised straightforwardly, as demonstrated below.

```
theorem exists_fixed_point (hT : is_contraction T q) (x₀ : X) [CompleteSpace X] :
    ∃ x' : X, T x' = x' := by
  -- The sequence generated by the contraction converges
  obtain ⟨x', hx'⟩ := seq_converges hT x₀
  -- The limit of the sequence is a fixed point
  exact ⟨x', limit_fixed hT x₀ hx'⟩
```

It remains to show that the fixed point is unique.

Theorem 2.16. A contraction has a unique fixed point.

Informal proof. Assume $T(x) = x$ and $T(y) = y$. Then

$$d(x, y) = d(T(x), T(y)) \leq qd(x, y),$$

where $0 \leq q < 1$. This implies

$$d(p, q)(1 - q) \leq 0.$$

Since $1 - q > 0$, it follows that $d(x, y) = 0$, hence $x = y$. Therefore, the fixed point is unique. □

```
theorem contraction_fixed_point_unique (hT : is_contraction T q) (x y : X)
  (hx : T x = x) (hy : T y = y) :
    x = y := by
  -- Apply the contraction property to the fixed points
  have h_contr := hT.2.2
  have h_dist : dist (T x) (T y) ≤ q * dist x y := h_contr x y
  -- Use the fixed point properties to simplify the distance
  rw [hx, hy] at h_dist
  -- Since ‘q < 1‘, we have ‘0 < 1 - q‘
  have h_one_minus_q_pos : 0 < 1 - q := one_minus_q_pos hT
  -- Multiply both sides by ‘1 - q‘ and conclude that the distance is zero
  have h_zero : (1 - q) * dist x y ≤ 0 := by linarith
  have h_dist_zero : dist x y = 0 := by
    apply le_antisymm
    · exact nonpos_of_mul_nonpos_right h_zero h_one_minus_q_pos
    · apply dist_nonneg
  -- Conclude that the fixed points are equal
  exact eq_of_dist_eq_zero h_dist_zero
```

2.5 The theorem

We can now state the main theorem of this project.

Theorem 2.17 (Banach Fixed Point Theorem [Lee03, Lemma C.35])**.** Let X be a non-empty complete metric space. Every contraction $T : X \to X$ has a fixed point.

Informal proof. The proof of this theorem is obtained by combining the results established earlier in this report. We begin by defining a sequence of iterates $x_n = T(x_{n-1})$, starting from an initial point $x_0 \in X$. By Theorem 2.11, this sequence is Cauchy, and since we are working in a complete metric space, it follows that the sequence converges to a limit. Lemma 2.15 establishes that this limit is a fixed point of the contraction mapping. Finally, Theorem 2.16 ensures that this fixed point is unique. □

```
theorem banach_fixed_point (hT : is_contraction T q) (x₀ : X) [CompleteSpace X] :
    ∃! x : X, T x = x := by
  -- Obtain an approximate fixed point ‘x’‘ using the existence theorem
  obtain ⟨x', hT_x'⟩ := exists_fixed_point hT x₀
  use x' -- Declare ‘x’‘ as the unique fixed point
  constructor -- Prove both existence and uniqueness
  · -- First, showing that ‘x’‘ is indeed a fixed point
    exact hT_x'
  · -- Next, proving uniqueness
    intro x hx
    -- Using the uniqueness result for contraction mappings
    exact Eq.symm (contraction_fixed_point_unique hT x' x hT_x' hx)
```

The formalisation of this theorem in Lean is concise, as its proof relies on the lemmas previously established.

2.6 The #lint test

At the conclusion of the code, the `#lint` command was executed, producing the following output:

```
-- Found 0 errors in 17 declarations (plus 21 automatically generated ones)
-- in the current file with 15 linters
-- All linting checks passed!
```

Note that, in contrast to the previous project, `#lint` initially produced errors such as `unusedArguments` and `unusedHavesSuffices`. As such, unnecessary statements and arguments were removed.

3 Challenges faced

It was surprising to find that implementing simple algebraic manipulations of inequalities in Lean proved to be unexpectedly challenging. For instance, the use of `/` often posed significant difficulties, causing tactics to fail in recognising the appropriate steps. Consequently, it was sometimes necessary to multiply by the reciprocal to guide Lean effectively.

A similar challenge arose when formalising the necessary inequalities for the proof of Theorem 2.11. This required the extensive use of cumbersome and unaesthetic `rw` tactics. Although functional, this approach lacked elegance and highlighted limitations in managing algebraic expressions within Lean.

Furthermore, working with the notion of convergence using `Filters` presented significant challenges, especially since this was attempted before the relevant lecture material was covered. This lack of background knowledge made the formalisation process more complex and time-consuming. Overall, this project was considerably more difficult than Project 1. A substantial amount of time and effort was invested to ensure that the formalisation was complete and `sorry`-free, sometimes at the expense of code style and elegance in the implementation.

4 Reflection

One potential improvement for this project would be to generalise the formalisation by using `EMetricSpace`‘ instead of `MetricSpace`. The current approach assumes all distances are finite, which simplifies the proofs of convergence and fixed points but limits the generality of the results. Using `EMetricSpace` would allow the formalisation to handle cases with potentially infinite distances, thus broadening the applicability of the contraction mapping theorem. However, this generalisation would introduce additional complexity, particularly in managing infinite limits and ensuring boundedness of sequences. Balancing this generality with the increased complexity would be a worthwhile consideration for future work.

From a stylistic perspective, better use of `variable` could have improved the readability and maintainability of the code. However, difficulties arose when attempting to declare certain variables, such as `(x_0 : X)`, leading to challenges in appropriately scoping them. Additionally, more frequent use of `open`, for example with `open Nat` and `open Filter`, could have streamlined the code by reducing verbosity. However, this approach led to numerous naming conflicts, and due to concerns about breaking the code, these changes were avoided. Although using fully qualified names like `namespace.theorem` improves clarity, it can reduce conciseness, which is a matter of personal preference. Furthermore, using `namespace` instead of `section` might have provided better logical organisation of the code. In future work, a more balanced approach to these stylistic choices would enhance the overall structure and readability of the formalisation.

References

[Ban22] Stefan Banach. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta Mathematicae*, 3(1):133–181, 1922.

[Cac23] R. Caccioppoli. *Un teorema generale sull'esistenza di elementi uniti in una trasformazione funzionale: nota*. Bardi, 1923.

[JJT24] Jacek Jachymski, Izabela Jóźwik, and Małgorzata Terepeta. The banach fixed point theorem: selected topics from its hundred-year history. *Revista de la Real Academia de Ciencias Exactas, Físicas y Naturales. Serie A. Matemáticas*, 118(4):140, 2024.

[Lee03] J.M. Lee. *Introduction to Smooth Manifolds*. Graduate Texts in Mathematics. Springer, 2003.

[Rho77] Billy Rhoades. A comparison of various definitions of contractive mappings. *Transactions of The American Mathematical Society - TRANS AMER MATH SOC*, 226:257–257, 02 1977.

[Sut75] W. A. (Wilson Alexander) Sutherland. *Introduction to metric and topological spaces*. Oxford University Press, Oxford, second edition. edition, 2009 – 1975.