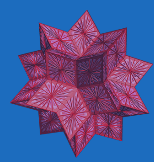
# Projeto Computacional
# 2.ª Parte

MATEMÁTICA



EXPERIMENTAL

RELATÓRIO

Trabalho realizado por:
Afonso Ribeiro, ist1102763
Diogo Rodrigues, ist1113787
Pedro Mendes, ist1109994

Com o apoio dos professores:
Pedro Lima
André Crispim

# 1. Frações contínuas e constante de Khinchin

## a)

- Esta função aproxima a constante de Khinchin ($\approx 2.68545$) através do produto infinito:

$$K_0 \;=\; \prod_{k=1}^{\infty}\left(1+\tfrac{1}{k(k+2)}\right)^{\log_2(k)}.$$

- O parâmetro `epsilon` especifica a tolerância para o erro absoluto.
- A cada iteração, calcula-se o próximo fator $\Big(1+1/[k(k+2)]\Big)^{\log_2(k)}$, multiplica-se ao produto parcial, e verifica-se se a diferença $|\texttt{produtoParcial} - K_0|$ é menor que $\epsilon$.
- Quando o `If` deteta que o erro ficou abaixo de $\epsilon$, a função retorna o valor do produto e o número de fatores $k$ necessários.

## b)

- Esta função calcula a constante de Khinchin a partir da fração contínua de um número real.
- A ideia baseia-se no limite:

$$K_0 \;=\; \lim_{n\to\infty}\Big(a_1 a_2 \cdots a_n\Big)^{1/n},$$

onde $a_1, a_2, \ldots$ são os coeficientes da fração contínua (ignorando o termo inicial $a_0$).
- A função extrai muitos coeficientes via `ContinuedFraction[x, ...]` e depois acumula o produto dos $a_i$, calcula a média geométrica, e compara-a com o valor de referência `Khinchin`.
- Se $|\texttt{geom} - K_0| < \epsilon$, retorna o valor e o número $n$ de coeficientes usados. Caso contrário, após esgotar o número máximo de coeficientes, imprime uma mensagem de erro.

## c)

- $\pi$: A chamada à função `khinchin2[`$\pi, \epsilon$`]`, para os valores de $\epsilon$ na lista $\{$`0.1, 0.01, 0.001, 0.0001`$\}$ devolveu valores sucessivamente mais próximos da constante de Khinchin.
- e: A chamada à função `khinchin2[e, 0.01]` devolveu um valor próximo da constante de Khinchin, mas a chamada `khinchin2[e, 0.001]` falhou.
- `ln(2)`: A chamada à função `khinchin2[ln(2), `$\epsilon$`]`, para os valores de $\epsilon$ na lista $\{$`0.1, 0.01, 0.001, 0.0001`$\}$ devolveu valores sucessivamente mais próximos da constante de Khinchin.
- $\frac{1+\sqrt{5}}{2}$: A chamada à função `khinchin2[`$\frac{1+\sqrt{5}}{2}$`, 0.001]` falhou.
- $\frac{\sqrt{3}+1}{2}$: A chamada à função `khinchin2[`$\frac{\sqrt{3}+1}{2}$`, 0.001]` falhou.
- $2^{\frac{1}{3}}$: A chamada à função `khinchin2[`$2^{\frac{1}{3}}, \epsilon$`]`, para os valores de $\epsilon$ na lista $\{$`0.1, 0.01, 0.001, 0.0001`$\}$ devolveu valores sucessivamente mais próximos da constante de Khinchin.
- $e^{\pi}$: A chamada à função `khinchin2[`$e^{\pi}, \epsilon$`]`, para os valores de $\epsilon$ na lista $\{$`0.1, 0.01, 0.001, 0.0001`$\}$ devolveu valores sucessivamente mais próximos da constante de Khinchin.

- `sin(1)`: A chamada à função `khinchin2[sin(1), ` $\epsilon$`]`, para os valores de $\epsilon$ na lista {`0.1, 0.01, 0.001, 0.0001`} devolveu valores sucessivamente mais próximos da constante de Khinchin.

- `tan(1/2)`: A chamada à função `khinchin2[tan(1/2), 0.1]` devolveu um valor próximo da constante de Khinchin, mas a chamada `khinchin2[tan(1/2), 0.01]` falhou.

Verifica-se, então, que, para os números $\pi$, `ln(2)`, $2^{\frac{1}{3}}$, $e^{\pi}$ e `sin(1)`, o limite converge para a constante de Khinchin. No entanto, era expectável que, para os valores e e `tan(1/2)`, o limite convergisse, formulando-se a conjetura de que somente para números racionais e irracionais quadráticos o limite não converge para a constante de Khinchin, e que para todos os demais (transcendentes e algébricos de grau $\geq 3$) converge. Estes dois valores de $x$ obtiveram um resultado próximo da constante pretendida para valores mais altos de $\epsilon$, falhando depois com valores mais baixos. Esta falha poderá estar relacionada com a necessidade de obter os resultados com um maior número de elementos nas frações contínuas dos números em questão, o que não nos foi possível, pois a computação da função `khinchin2` não terminava em tempo útil nessas condições.

## d)

Para cada um dos valores de $\epsilon$ especificados, os valores de $k_{max}$ para o comando `khinchin1` e de $n_{max}$ para o comando `khinchin2` com $x = \pi$ e $x = \sin(1)$ foram, respetivamente, os seguintes.

| $\epsilon$ | $k_{\max}$ | $n_{\max}(x = \pi)$ | $n_{\max}(x = \sin(1))$ |
|---|---|---|---|
| $10^{-1}$ | 246 | 16 | 4 |
| $10^{-2}$ | 3547 | 17 | 57 |
| $10^{-3}$ | 45411 | 117 | 327 |
| $10^{-4}$ | 550885 | 976 | 1627 |

Discussão da eficiência dos métodos utilizados:

- **Método `khinchin1` (produto infinito)**: É um método universal, isto é, não depende de um número $x$: uma vez implementado, aproxima a constante de Khinchin sem nenhuma informação adicional. Podemos observar na tabela que, para alcançar precisões como $10^{-3}$ ou $10^{-4}$, o número de fatores $k_{\max}$ se torna muito grande ($45411$ e $550885$, respectivamente). Portanto, em termos de iterações, este método cresce de maneira expressiva conforme $\epsilon$ diminui.

- **Método `khinchin2` (frações contínuas)**: Em vários casos (nomeadamente $\pi$ para $\epsilon = 10^{-1}$ e $\epsilon = 10^{-2}$, ou $\sin(1)$ para $\epsilon = 10^{-1}$), o número de coeficientes necessários ($n_{\max}$) foi bastante menor do que o $k_{\max}$ do produto infinito para a mesma tolerância. No entanto, este método pode convergir mais devagar, dependendo da distribuição estatística dos coeficientes $a_n$ do $x$ escolhido, como pudemos constatar ao efetuar as computações no `Mathematica`. Para além disso, este método pode falhar rapidamente se $x$ for irracional quadrático (pois não converge para a constante), ou se o limite de iterações não for suficiente no caso de convergência lenta, uma desvantagem relativamente ao primeiro método, que não falha.

Assim, conclui-se que existe um trade-off entre a universalidade e simplicidade do método do produto infinito e a potencial rapidez (mas dependente de uma boa escolha de $x$) do método das frações contínuas.

## 2.Raíz Digital

## Alínea a)

**Prove (analiticamente) que p(m+9) = p(m), para todo o m natural**

p(1) = 1

p(1+9) = p(10) = p(1) =1

Hipótese de Indução: p(m) = p(m+9)

Tese: p(m+1) = p((m+9)+1) ⇔ p(m+1) = p(m+10)

Seja m = $d_k \times 10^k + d_{k-1} \times 10^{k-1} + \ldots + d_1 \times 10 + d_0$, onde $d_i$ são os dígitos de m

Representando m+1 e m+10 na base decimal, tem-se que:

$$m+1 = d_k \times 10^k + d_{k-1} \times 10^{k-1} + \ldots + d_1 \times 10 + d_0 + 1$$

$$m+10 = d_k \times 10^k + d_{k-1} \times 10^{k-1} + \ldots + (d_1 \times 10 + 10) + d_0$$

Ao adicionar 10 ao número m, o algarismo das dezenas de m é incrementado de 1 unidade. Ao adicionar 1 ao número m, o algarismo das unidades de m é incrementado de 1 unidade.

Se o algarismo das unidades de m for 9, o algarismo das unidades de m + 1 é 0. O algarismo das dezenas é incrementado de 1 unidade.
Se o algarismo das dezenas de m for 9, o algarismo das dezenas de m + 10 é 0. O algarismo das centenas é incrementado de 1 unidade.

Logo, a soma dos dígitos de m+10 é igual à soma dos dígitos de m+1.

Seja S(m) a soma dos dígitos de m tem-se, portanto:

$$S(m+1) = S(m+10)$$

Uma vez que a raiz digital de m se obtém adicionado, sucessivamente, os seus dígitos, tem-se que S(m+1) = S(m+10) e, consequentemente, p(m+1) = p(m+10).

Tendo provado a tese, concluímos que p(m) = p(m+9).

**Prove (analiticamente) que se m não é múltiplo de 9, então p(m) = mod(m,9)**

Seja m = $d_k \times 10^k + d_{k-1} \times 10^{k-1} + \ldots + d_1 \times 10 + d_0$, onde $d_i$ são os dígitos de m

Sabemos que $10^n \equiv 1 \bmod 9$, com $n \in \mathbb{N}$

Então, m mod 9 = $(d_k \times 1 + d_{k-1} \times 1 + \ldots + d_1 \times 1 + d_0)$ mod 9

ou seja, m mod 9 = $(d_k + d_{k-1} + \ldots + d_1 + d_0)$ mod 9

Se m não é múltiplo de 9, então m mod 9 ≠ 0. Logo, p(m) ≡ m mod 9.

Ou seja, p(m) = mod (m, 9)

**Prove (analiticamente) que se m é múltiplo de 9, então p(m) = 9**

Seja m = $d_k \times 10^k + d_{k-1} \times 10^{k-1} + \ldots + d_1 \times 10 + d_0$, onde $d_i$ são os dígitos de m

Sabemos que $10^n \equiv 1 \bmod 9$, com $n \in \mathbb{N}$

E, consequentemente, $m \equiv (d_k + d_{k-1} + \ldots + d_1 + d_0) \bmod 9$

Como m é múltiplo de 9, $m \equiv 0 \bmod 9$

Sabemos ainda que $p(m) \equiv m \bmod 9$. Logo, $p(m) \equiv 0 \bmod 9$.

Como p(m) é um único dígito decimal, p(m) = 0 ou p(m) = 9. Como p(m) é um número inteiro positivo, o único valor possível de p(m) é 9.

Logo, p(m) = 9.

## Alínea b)

$$\text{DigitalRoot}[\text{n\_Integer}] := \text{NestWhile}[\text{DigitSum}, n, \text{Function}[x, \text{IntegerLength}[x] > 1]]$$

A função **DigitalRoot** recebe como argumento um número inteiro e calcula a respetiva raíz digital. Partindo do número introduzido, a função DigitSum, que calcula a soma dos dígitos do número, é iterada sucessivas vezes até que a condição $\text{IntegerLength}[x] > 1]]$ seja falsa, isto é, até que o número obtido seja constituído por um único dígito, ou seja, até obter a raíz digital do número inserido.

## Alínea c)

**Alínea a)**

$$\text{Map}[\text{DigitalRoot}, \text{Range}[100]]$$

O comando $\mathbf{Map}[\mathbf{DigitalRoot}, \mathbf{Range}[\mathbf{100}]]$ aplica a função DigitalRoot a cada um dos elementos da lista dos primeiros 100 números inteiros, obtida através do comando Range[100], devolvendo uma lista com a raíz digital de cada um dos primeiros 100 números inteiros positivos.

**Alínea b)**

$$\text{First100Prime} := \text{Map}[\text{Prime}, \text{Range}[100]]$$

A função **First100Prime** aplica a função Prime, que calcula o i-ésimo número primo, a cada um dos elementos da lista dos primeiros 100 números inteiros positivos, obtida através do comando Range[100], devolvendo a lista com os primeiros 100 números primos.

$$\mathrm{Map}[\mathrm{DigitalRoot}, \mathrm{First100Prime}]$$

O comando $\mathbf{Map}[\mathbf{DigitalRoot}, \mathbf{First100Prime}]$ aplica a função DigitalRoot a cada um dos elementos da lista Firs100Prime, devolvendo uma lista com a raíz digital de cada um dos primeiros 100 números primos.

**Alínea c)**

$$\mathrm{First100Fibonacci} := \mathrm{Table}[\mathrm{Fibonacci}[n], \{n, 100\}]$$

A função **First100Fibonnaci** calcula os primeiros 100 números de Fibonacci e devolve a respetiva lista ordenada.

$$\mathrm{Map}[\mathrm{DigitalRoot}, \mathrm{First100Fibonacci}]$$

O comando $\mathbf{Map}[\mathbf{DigitalRoot}, \mathbf{First100Fibonacci}]$ aplica a função DigitalRoot a cada um dos elementos da lista First100Fibonacci, ou seja, calcula a raiz digital de cada um dos primeiros 100 números de Fibonacci e devolve a respetiva lista.

**Alínea d)**

$$\mathrm{FermatFunction}[\mathrm{n\_Integer}] := 2\text{^}2\text{^}n$$

A função **FermatFunction** recebe como argumento um número inteiro n e calcula o (n+1)-ésimo número de Fermat definido pela fórmula $(Fe)_n$ =2^2^n com n = 0, 1…

$$\mathrm{First15Fermat} := \mathrm{Map}[\mathrm{FermatFunction}, \mathrm{Range}[0,14]]$$

A função **First15Fermat** aplica a função FermatFunction a cada um dos elementos da lista dos primeiros 15 números inteiros não negativos, ou seja, calcula os primeiros 15 números de Fermat e devolve a respetiva lista.

$$\mathrm{Map}[\mathrm{DigitalRoot}, \mathrm{First15Fermat}]$$

O comando $\mathbf{Map}[\mathbf{DigitalRoot}, \mathbf{First15Fermat}]$ aplica a função DigitalRoot a cada um dos elementos da lista First15Fermat, ou seja, calcula a raiz digital de cada um dos primeiros 15 números de Fermat e devolve a respetiva lista.

**Alínea e)**

$$\text{DenomFirst200ConvPi} := \text{Denominator}[\text{Convergents}[\text{Pi}, 200]]$$

A função **DenomFirst200ConvPi** devolve a lista dos denominadores dos primeiros 200 convergentes da fração contínua do número Pi.

$$\text{Map}[\text{DigitalRoot}, \text{DenomFirst200ConvPi}]$$

O comando $\mathbf{Map}[\mathbf{DigitalRoot}, \mathbf{DenomFirst200ConvPi}]$ aplica a função DigitalRoot a cada um dos elementos da lista DenomFirst200ConvPi, ou seja, calcula a raiz digital de cada um dos denominadores dos primeiros 200 convergentes da função contínua do número Pi e devolve a respetiva lista.

**Alínea f)**

$$\text{DenomFirst200ConvNeper} \coloneqq \text{Denominator}\big[\text{Convergents}[\text{E}, 200]\big]$$

A função **DenomFirst200ConvNeper** devolve a lista dos denominadores dos primeiros 200 convergentes da fração contínua do número e.

$$\text{Map}[\text{DigitalRoot}, \text{DenomFirst200ConvNeper}]$$

O comando $\mathbf{Map}[\mathbf{DigitalRoot}, \mathbf{DenomFirst200ConvNeper}]$ aplica a função DigitalRoot a cada um dos elementos da lista DenomFrist200ConvNeper, ou seja, calcula a raiz digital de cada um dos denominadores dos primeiros 200 convergentes da função contínua do número Pi e devolve a respetiva lista.

## Alínea d)

**1)**

$$\text{ListPlot}[\text{Map}[\text{DigitalRoot}, \text{Range}[100]], \text{Frame}{-}{>}\ \text{True}, \text{FrameLabel}{-}$$
$$>\{\text{"Ordem dos Números Inteiros Positivos"}, \text{"Raiz Digital"}\}, \text{PlotMarkers}{-}{>}\ \text{"●"}]$$

O comando acima constrói um gráfico da lista da raiz digital dos primeiros 100 números inteiros calculados anteriormente, $\text{Map}[\text{DigitalRoot}, \text{Range}[100]]$, que estabelece a relação entre a ordem dos números inteiros positivos (eixo das abcissas) e a raiz digital desses números (eixo das ordenadas). De modo a obter um gráfico detalhado o comando ListPlot incorpora ainda os seguintes comandos:

$\text{Frame}{-}{>}\ \text{True}$ - Adiciona uma moldura ao redor do gráfico para o conseguir legendar.

$\text{FrameLabel}{-}{>}\ \{\text{"Ordem dos Números Inteiros Positivos"}, \text{"Raiz Digital"}\}$ – Legenda o eixo das abcissas com o nome “Ordem dos Números Inteiros Positivos” e o eixo das ordenadas com o nome “Raiz Digital”.

$\text{PlotMarkers}{-}{>}\ \text{"●"}]$ – Usa círculos preenchidos “●” para assinalar os pontos do gráfico.

**2)**

$$\text{ListOccPosInt}: = \text{Map}[\text{Function}[x, \text{Count}[\text{Map}[\text{DigitalRoot}, \text{Range}[100]], x]], \text{Range}[9]]$$

A função **ListOccPosInt** aplica a função $\text{Function}[x, \text{Count}[\text{Map}[\text{DigitalRoot}, \text{Range}[100]], x]]$ a cada valor x pertencente ao conjunto {1,2,3,4,5,6,7,8,9}, Range[9] contando o número de vezes que x aparece na lista $\text{Map}[\text{DigitalRoot}, \text{Range}[100]]$, ou seja, dos 100 primeiros números inteiros, determina quantos têm raiz digital x.

$$\text{DataPositiveInteger}: = \text{Transpose}[\{\text{Range}[9], \text{ListOccPosInt}\}]$$

A função DataPositiveInteger combina a lista dos números inteiros positivos de 1 a 9, Range[9], e a lista ListOccPosInt em uma lista de pares ordenados, ou seja, devolve uma lista de sublistas, cada uma constituída por uma raiz digital e pela respetiva quantidade de números inteiros que têm essa raiz. Na lista, as raízes digitais encontram-se por ordem crescente.

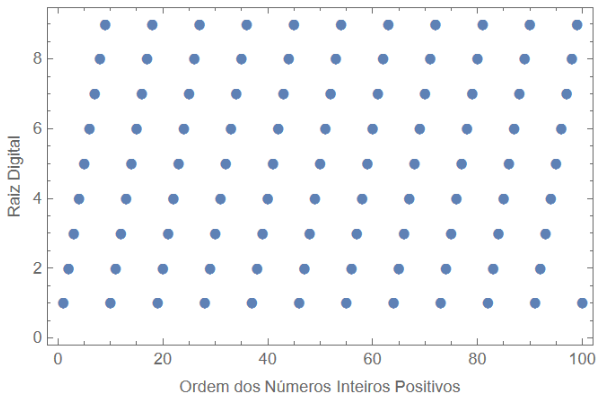
$$\text{Grid}[\text{Prepend}[\text{DataPositiveInteger}, \{\text{"Dígito"}, \text{"Número de Ocorrências"}\}], \text{Frame}{-}{>}\ \text{All}]$$

O comando Prepend adiciona a lista $\{\text{"Dígito"}, \text{"Número de Ocurrências"}\}$ ao início da lista DataPositiveInteger, servindo como cabeçalho da tabela. Estes dados são organizados numa tabela recorrendo ao comando Grid. O argumento $\text{Frame}{-}{>}\ \text{All}$ adiciona a linha que contorna todas as células da tabela.

Em todos os outros casos (alíneas b), c), d), e), f), segue-se o mesmo processo efetuando-se os ajustes necessários em relação aos argumentos.

**Análise dos resultados**

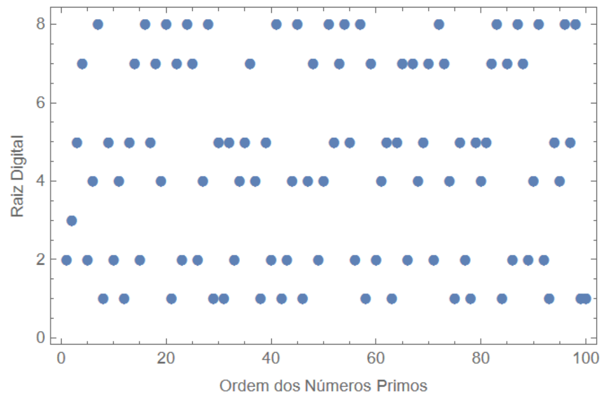
**Alínea a)**




| Dígito | Número de Ocorrências |
|---|---|
| 1 | 12 |
| 2 | 11 |
| 3 | 11 |
| 4 | 11 |
| 5 | 11 |
| 6 | 11 |
| 7 | 11 |
| 8 | 11 |
| 9 | 11 |

Existe um padrão cíclico na sequência das raízes digitais dos primeiros 100 números inteiros. A sequência 1, 2, 3, 4, 5, 6, 7, 8, 9 repete-se ao longo da lista das raízes digitais dos primeiros 100 números inteiros, tal facto justifica-se pela propriedades p(m) = p(m+9) e p(m) ≡ m mod 9, demonstradas anteriomente. Como p(m) ≡ m mod 9, podemos obter todos os números que têm de raiz digital 1, 2, 3, 4, 5, 6, 7, 8 ou 9. Todos os números que têm raiz digital n com n pertencente ao conjunto {1,2,3,4,5,6,7,8,9} são dados pela expressão n + 9q, sendo q um número inteiro não negativo. Assim conclui-se que a raiz digital se repete de 9 em 9 números inteiros. Verificamos ainda que todos os dígitos têm a mesma frequência, exceto o número 1 que tem mais uma ocorrência, uma vez que também se procede ao cálculo da raiz digital de 100 (o ciclo que se inicia no centésimo número inteiro está incompleto).

**Alínea b)**




| Dígito | Número de Ocorrências |
|---|---|
| 1 | 16 |
| 2 | 18 |
| 3 | 1 |
| 4 | 15 |
| 5 | 18 |
| 6 | 0 |
| 7 | 16 |
| 8 | 16 |
| 9 | 0 |

Não existe nenhum padrão cíclico na sequência das raízes digitais dos primeiros 100 números primos, tal facto poder-se-á justificar por não existir um padrão cíclico para determinar uma dada sequência de números primos.

Verifica-se ainda que os dígitos 6, 9 e 3 (exceto para o número primo 3) não aparecem na sequência das raízes digitais dos primeiros 100 números primos, o que se justica pela propriedade $p(m) \equiv m \bmod 9$, demonstrada anteriormente. Para que 3, 6 ou 9 sejam raízes digitais de um dado número m, ter-se-á de ter:

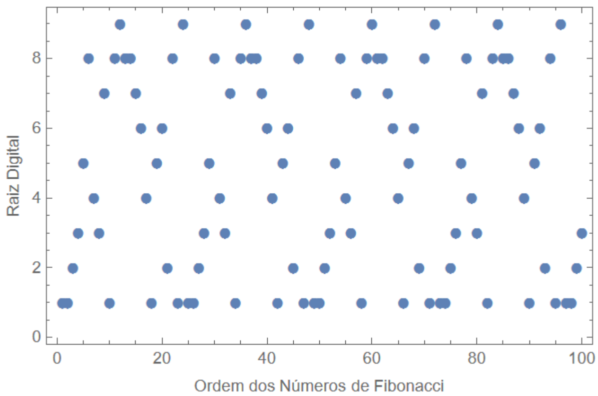
Raiz digital 3: $3 \equiv m \bmod 9 \Leftrightarrow m = 3 + 9q$ com q inteiro não negativo

Raiz digital 6: $6 \equiv m \bmod 9 \Leftrightarrow m = 6 + 9q$ com q inteiro não negativo

Raiz digital 9: $9 \equiv m \bmod 9 \Leftrightarrow m = 9q$ com q inteiro não negativo

Ou seja, para que 3, 6 ou 9 fosse raiz digital de m, m teria de ser divísivel por 3. Além disso, para que 9 fosse raiz digital de m, m teria ainda de ser divisível por 9. Como m é primo, m só é divisível por 3 se m = 3, o que explica porque é que 3 é raiz digital de apenas um número. Assim, conclui-se que 3 só é raiz digital para 3 e que 6 e 9 não são raízes digitais de m.

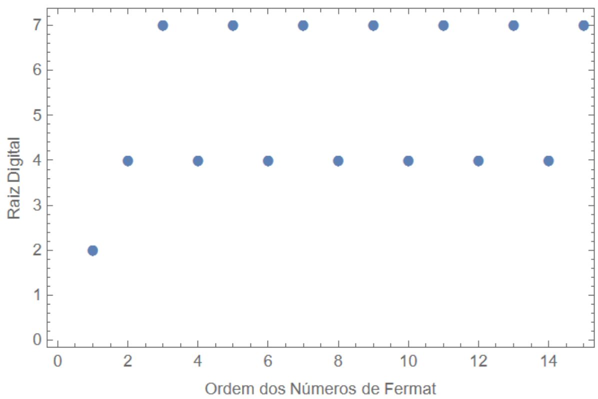
**Alínea c)**




| Dígito | Número de Ocorrências |
|---|---|
| 1 | 22 |
| 2 | 9 |
| 3 | 9 |
| 4 | 8 |
| 5 | 8 |
| 6 | 8 |
| 7 | 8 |
| 8 | 20 |
| 9 | 8 |

Existe um padrão cíclico na sequência das raízes digitais dos primeiros 100 números de Fibonacci. A sequência 1,1,2,3,5,8,4,3,7,1,8,9,8,8,7,6,4,1,5,6,2,8,1,9 repete-se ao longo da lista das raízes digitais dos primeiros 100 números de Fibonacci, tal facto justifica-se aplicando a fórmula $F_{n+2} = F_{n+1} + F_n$ mod 9 para diversos valores de n e confirmamos pelos cálculos que os restos 1 e 8 são bastante mais comuns, pelo que os dígitos 1 e 8 tiveram muito mais ocorrências do que os restantes dígitos. Os restantes números aparecem com mais ou menos a mesma frequência.

**Alínea d)**




| Dígito | Número de Ocorrências |
|---|---|
| 1 | 0 |
| 2 | 1 |
| 3 | 0 |
| 4 | 7 |
| 5 | 0 |
| 6 | 0 |
| 7 | 7 |
| 8 | 0 |
| 9 | 0 |

Verifica-se que os dígitos 1, 3, 5, 6, 8 e 9 não aparecem na sequência das raízes digitais dos primeiros 15 números de Fermat definido pela fórmula $(Fe)_n$ =2^2^n com n = 0, 1... Além disso, existe um padrão cíclico na sequência das raízes digitais dos primeiros 15 números de Fermat. A sequência 4,7 repete-se ao longo da lista das raízes digitais dos primeiros 15 números de Fermat após o primeiro aparecimento da raiz digital 2, tal facto justifica-se pela propriedades p(m) ≡ m mod 9, demonstradas anteriomente. Como p(m) ≡ m mod 9, tem-se que:

$p(m) \equiv (Fe)_n \bmod 9 \Leftrightarrow 2^{2^n} \equiv 4 \bmod 9 \vee 2^{2^n} \equiv 7 \bmod 9 \Leftrightarrow$

$\Leftrightarrow 2^{2^n} \equiv 2^2 \bmod 9 \vee 2^{2^n} \equiv -2 \bmod 9 \Leftrightarrow$

$\Leftrightarrow 2^n \equiv 2 \bmod 6 \vee 2^{2^n} \equiv 2^4 \bmod 9 \Leftrightarrow$

$\Leftrightarrow 2^{(n-1)} \equiv 1 \bmod 3 \vee 2^n \equiv 4 \bmod 6 \Leftrightarrow$

$\Leftrightarrow 2^{(n-1)} \equiv 2^2 \bmod 3 \vee 2^{(n-1)} \equiv 2 \bmod 3 \Leftrightarrow$

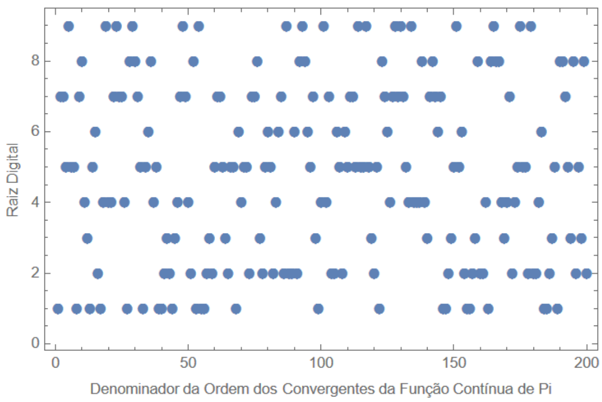
$\Leftrightarrow n-1 \equiv 2 \bmod 2 \vee n-1 \equiv 1 \bmod 2 \Leftrightarrow$

$\Leftrightarrow n \equiv 1 \bmod 2 \vee n \equiv 0 \bmod 2$

Conclui-se que se $n \equiv 1 \bmod 2$, $4 \equiv (Fe)_n \bmod 9$ e se $n \equiv 0 \bmod 2$, $7 \equiv (Fe)_n \bmod 9$, ou seja, se n for ímpar a raiz digital do respetivo número de Fermat é 7 enquanto que se n for par a raiz digital do respetivo número de Fermat é 4. Como consideramos o primeiro número de Fermat o número definido por $(Fe)_0 = 2^{2^0} = 2$, tem-se que se a ordem do número de Fermat for ímpar, a raíz digital é 4 enquanto que se a ordem for par, a raíz digital é 7.

Assim, conclui-se porque é que os números têm a mesma frequência e os restantes números (à exceção do 2 que é raiz digital do primeiro número de Fermat) não aparecem uma única vez.

**Alínea e)**



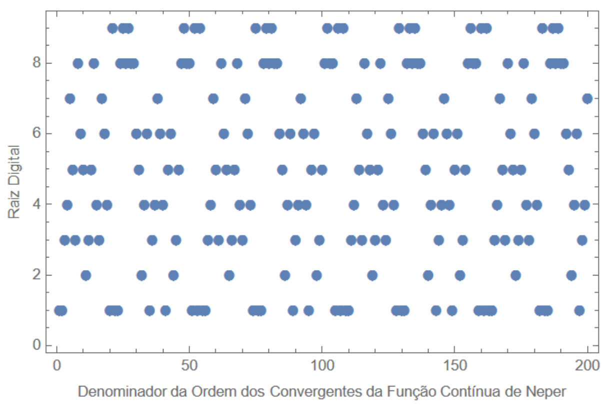

| Dígito | Número de Ocorrências |
|---|---|
| 1 | 23 |
| 2 | 30 |
| 3 | 15 |
| 4 | 23 |
| 5 | 32 |
| 6 | 13 |
| 7 | 27 |
| 8 | 19 |
| 9 | 18 |

Os dígitos 5 e 7 foram os que apareceram mais vezes e os dígitos 3 e 6 foram os que apareceram menos vezes. Calculando a fração contínua de $\pi$ para os primeiros 200 números, obtemos a seguinte lista:

{3, 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, 14, 2, 1, 1, 2, 2, 2, 2, 1, 84, 2, 1, 1, 15, 3, 13, 1, 4, 2, 6, 6, 99, 1, 2, 2, 6, 3, 5, 1, 1, 6, 8, 1, 7, 1, 2, 3, 7, 1, 2, 1, 1, 12, 1, 1, 1, 3, 1, 1, 8, 1, 1, 2, 1, 6, 1, 1, 5, 2, 2, 3, 1, 2, 4, 4, 16, 1, 161, 45, 1, 22, 1, 2, 2, 1, 4, 1, 2, 24, 1, 2, 1, 3, 1, 2, 1, 1, 10, 2, 5, 4, 1, 2, 2, 8, 1, 5, 2, 2, 26, 1, 4, 1, 1, 8, 2, 42, 2, 1, 7, 3, 3, 1, 1, 7, 2, 4, 9, 7, 2, 3, 1, 57, 1, 18, 1, 9, 19, 1, 2, 18, 1, 3, 7, 30, 1, 1, 1, 3, 3, 3, 1, 2, 8, 1, 1, 2, 1, 15, 1, 2, 13, 1, 2, 1, 4, 1, 12, 1, 1, 3, 3, 28, 1, 10, 3, 2, 20, 1, 1, 1, 1, 4, 1, 1, 1, 5, 3, 2, 1, 6, 1, 4, 1, 120, 2, 1, 1}

Observando esta lista, é seguro afirmar que todos estes números são completamente aleatórios, o que faz sentido pois π é um número irracional com infinitas casas decimais sem qualquer regularidade. Portanto, concluímos que não existe nenhum padrão cíclico na sequência das raízes digitais dos denominadores dos primeiros 200 convergentes da função contínua do número Pi

**Alínea f)**




| Dígito | Número de Ocorrências |
|---|---|
| 1 | 33 |
| 2 | 11 |
| 3 | 23 |
| 4 | 23 |
| 5 | 22 |
| 6 | 22 |
| 7 | 12 |
| 8 | 33 |
| 9 | 21 |

Os dígitos 1 e 8 são os que têm maior frequência, enquanto que 2 e 7 têm as menores ocorrências. Calculando a fração contínua de *e* para os primeiros 200 números, obtemos a seguinte lista:

{2, 1, 2, 1, 1, 4, 1, 1, 6, 1, 1, 8, 1, 1, 10, 1, 1, 12, 1, 1, 14, 1, 1, 16, 1, 1, 18, 1, 1, 20, 1, 1, 22, 1, 1, 24, 1, 1, 26, 1, 1, 28, 1, 1, 30, 1, 1, 32, 1, 1, 34, 1, 1, 36, 1, 1, 38, 1, 1, 40, 1, 1, 42, 1, 1, 44, 1, 1, 46, 1, 1, 48, 1, 1, 50, 1, 1, 52, 1, 1, 54, 1, 1, 56, 1, 1, 58, 1, 1, 60, 1, 1, 62, 1, 1, 64, 1, 1, 66, 1, 1, 68, 1, 1, 70, 1, 1, 72, 1, 1, 74, 1, 1, 76, 1, 1, 78, 1, 1, 80, 1, 1, 82, 1, 1, 84, 1, 1, 86, 1, 1, 88, 1, 1, 90, 1, 1, 92, 1, 1, 94, 1, 1, 96, 1, 1, 98, 1, 1, 100, 1, 1, 102, 1, 1, 104, 1, 1, 106, 1, 1, 108, 1, 1, 110, 1, 1, 112, 1, 1, 114, 1, 1, 116, 1, 1, 118, 1, 1, 120, 1, 1, 122, 1, 1, 124, 1, 1, 126, 1, 1, 128, 1, 1, 130, 1, 1, 132, 1, 1}

É possível observar um padrão nesta lista: em cada 3 elementos consecutivos, por 11, adiciona-se 2 ao terceiro número. Esta ocorrência pode explicar o facto de uns dígitos apareceram consideravelmente mais (ou menos) vezes do que outros. Contudo, ainda existe um fator aleatório e é difícil prever as raízes digitais sem experimentação. Como a lista acima satisfaz uma recorrência linear, p(m) segue uma recorrência cíclica, o que explica o padrão cíclico existente na sequência das raízes digitais dos denominadores dos primeiros 200 convergentes da função contínua e.