Física Computacional I

Dr. Roberto Pedro Duarte Zamorano

Universidad de Sonora

Temario

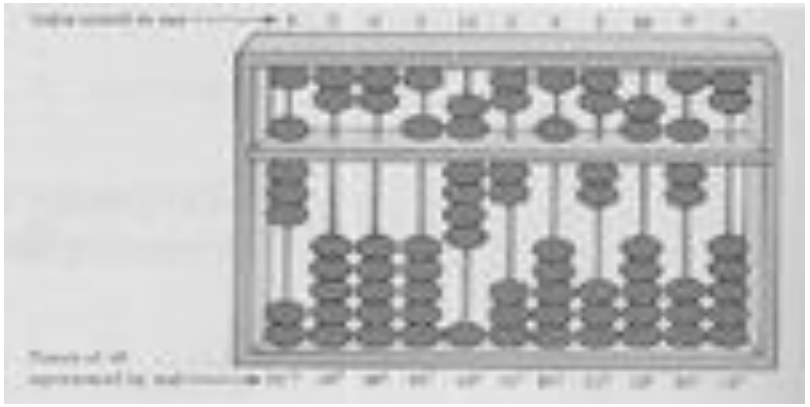
1. **Introducción.**
2. Problemas de valor inicial para ecuaciones diferenciales.
3. Métodos de Euler y de Taylor de orden superior.
4. Métodos de Runge-Kutta.
5. Ecuaciones de orden superior y sistemas de ecuaciones diferenciales.
6. Método de disparo lineal y método de disparo para problemas no lineales.
7. Sistemas de ecuaciones lineales y estrategias de pivoteo.
8. Valores y vectores propios de matrices.

Introducción.

1. Origen y apuntes históricos de la computación.
2. Ambiente GNU/Linux.
   a) Comandos más utilizados.
   b) Aplicaciones varias: Editores, Graficadores, Compiladores, Computación especializada.
3. Elementos básicos de programación FORTRAN.
4. Solución numérica de ecuaciones de una variable.

Los primeros orígenes.

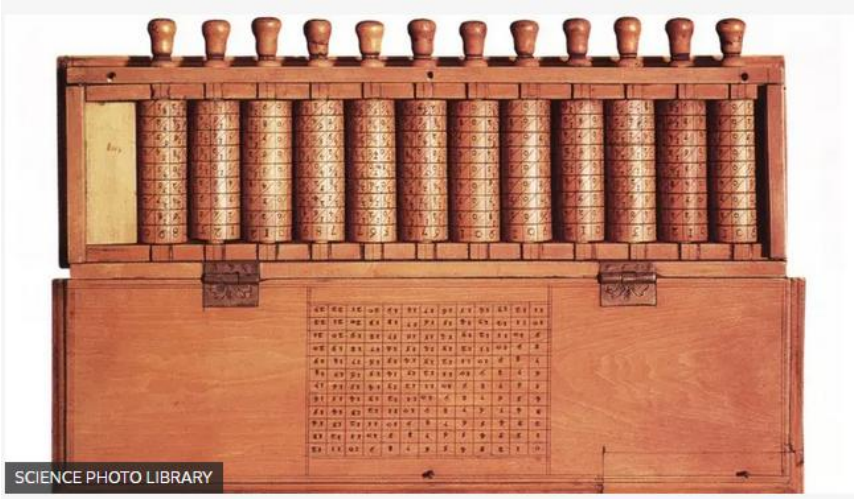
- La necesidad de calcular impulsó el desarrollo del ábaco como primera calculadora.



- Muhammad ibn Musa Al'khowarizmi (S. XII) desarrolló el concepto de un proceso a seguir con la finalidad de alcanzar un objetivo, lo que conocemos como algoritmo.

Los primeros orígenes. Siglo XVII

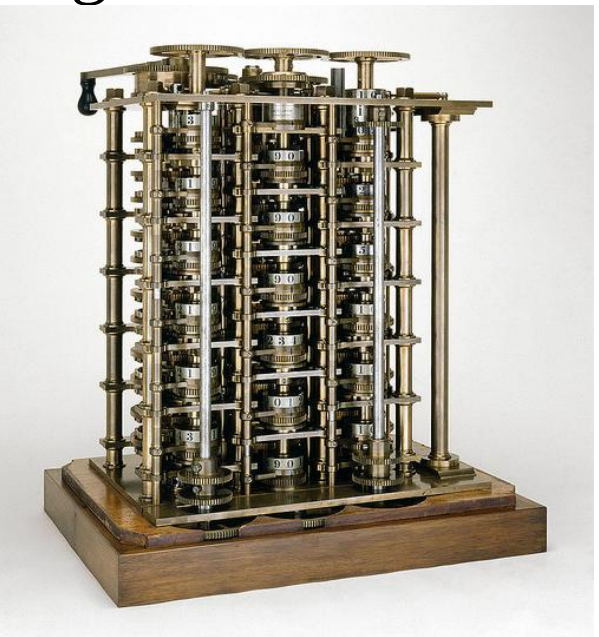
- 1612, John Napier hizo el primer uso impreso del punto decimal. Creó los logaritmos y varias máquinas para multiplicar.
- 1622, William Oughtred creó la regla de cálculo.
- 1642, Blaise Pascal creó una máquina de sumar.



Los primeros orígenes. Siglo XIX

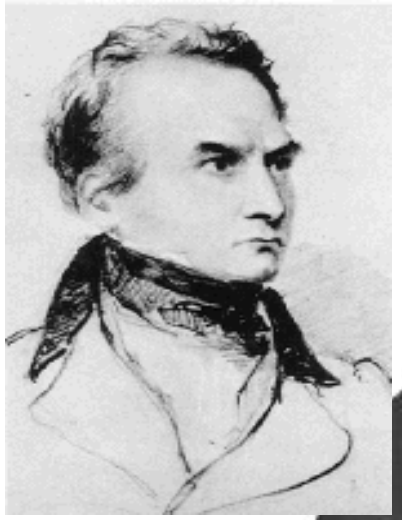
- 1801, Joseph-Marie Jaquard inventó un telar automático utilizando tarjetas perforadas.

- 1822, Charles Babbage diseñó una máquina diferencial para mejorar los cálculos de las tablas de navegación.



Los primeros orígenes. Siglo XIX

- 1833, Babbage abandona su proyecto inicial por considerarlo demasiado especializado y diseña la máquina analítica.



- 1842, Ada Augusta King, Lady Lovelace (hija de Lord Byron) colabora con Babbage considerándose la primera programadora.

Los primeros orígenes. Siglo XIX

- Problemas en el recuento del censo de Población USA en 1890, provocaron que Herman Hollerith desarrollase una máquina tabuladora, creando una empresa que en 1924 pasó a llamarse IBM.

Los primeros orígenes. 1925

- Vannevar Bush del Instituto Tecnológico de Massachusetts (MIT, por sus siglas en inglés) construyó una máquina diferencial con capacidades de integración y de diferenciación.

Los primeros orígenes. 1935-1938

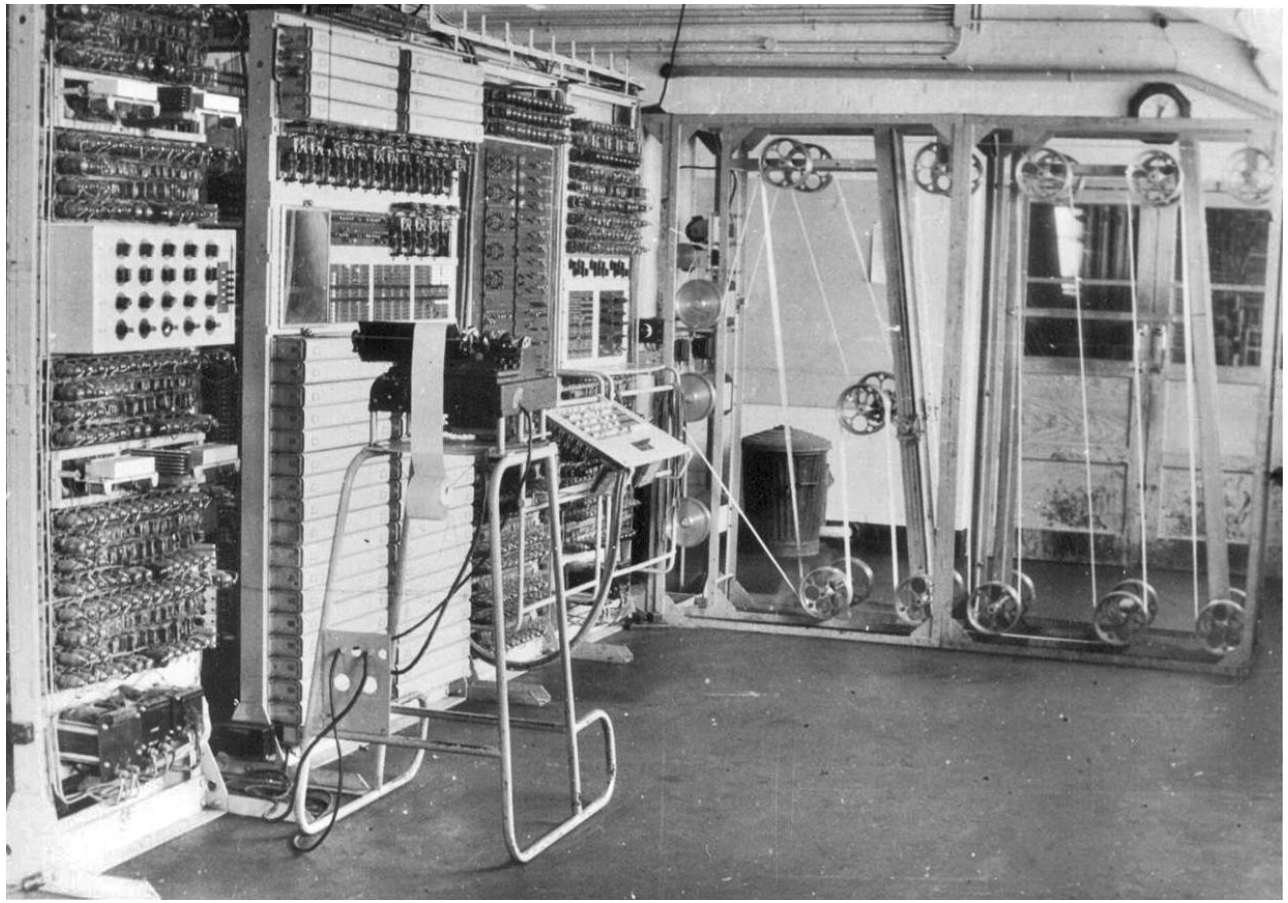
- Konrad Zuse, desarrolló el computador Z-1 utilizando relés (relevadores) y aritmética binaria.
- Después de la segunda guerra mundial, se instaló en Suiza, desarrolló el Z-4 y fundó una compañía de computadores absorbida posteriormente por Siemens.

Los orígenes modernos. 1937

- Alan Turing desarrolló la idea de máquina universal capaz de ejecutar cualquier algoritmo y formando la base de la computabilidad.
- Por sus aportes a la teoría de la computación, a Alan Turing se le reconoce como uno de los fundadores de la Computación moderna.

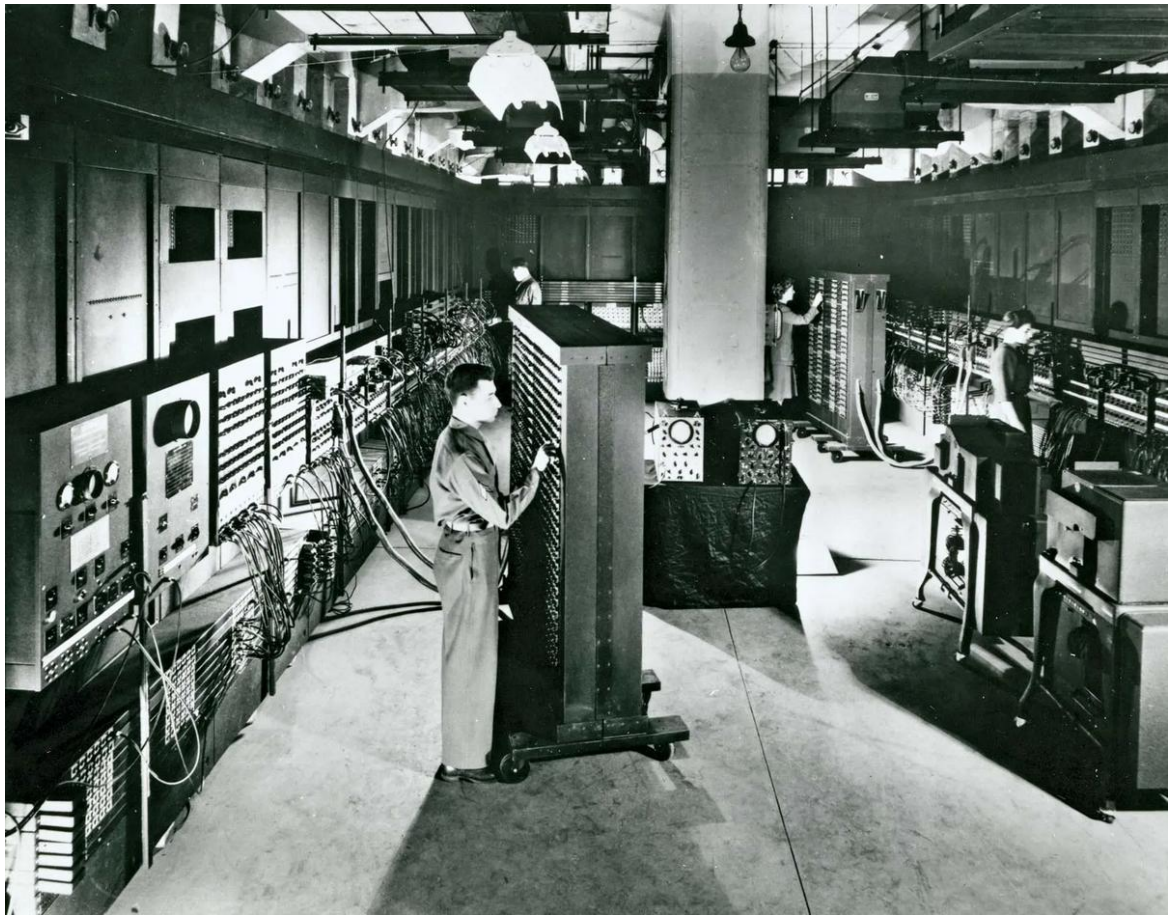
Los orígenes modernos. 1940-1944

- Durante la segunda guerra mundial, y dada la necesidad de descifrar los mensajes encriptados de los alemanes, se desarrolló, con la ayuda de Turing, la máquina denominada Colossus.



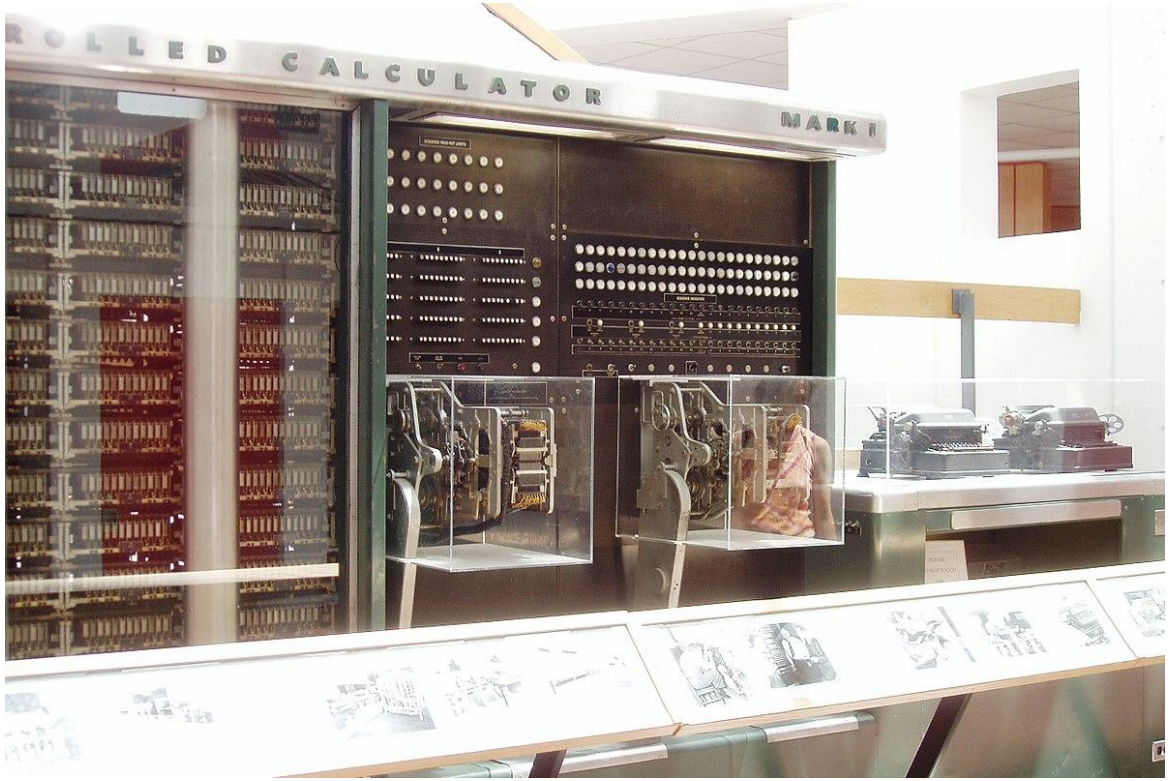
Los orígenes modernos. 1943

- Comenzó este año, en la Moore School of Electrical Engineering en Pennsylvania, el trabajo en el ENIAC (Electronic an Numeric Integrator and Computer)



Los orígenes modernos. 1944

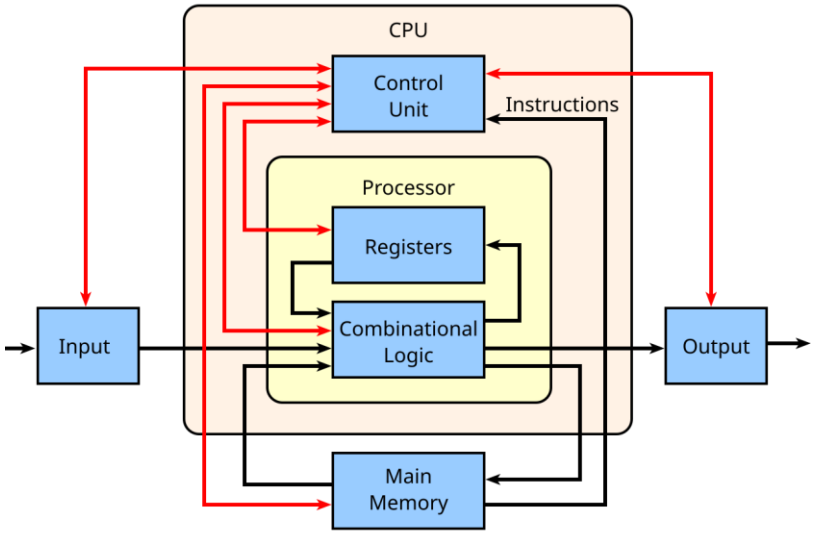
- El primer calculador electromecánico automático de propósito general a gran escala fue el Mark I, ideado por Howard Aiken y desarrollado por IBM.



Los orígenes modernos. 1945

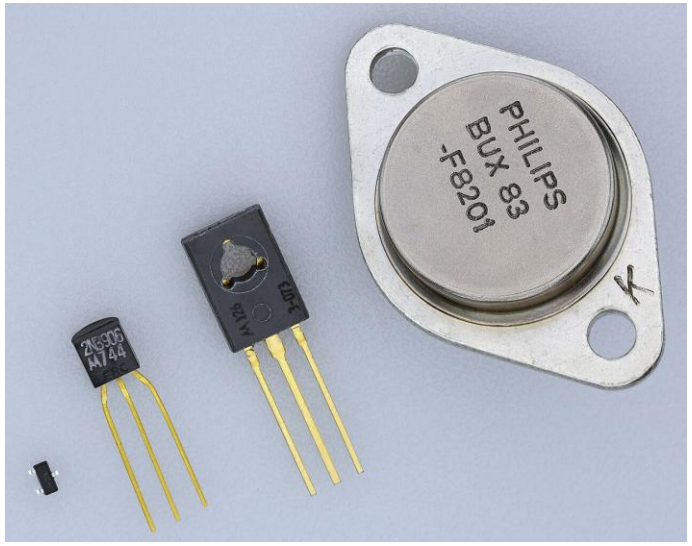
- Grace Murray Hopper encontró el primer "bug" en el computador Mark II.

- John von Neumann escribe su borrador sobre la arquitectura de los computadores el cual sirve de base al concepto actual de computador.



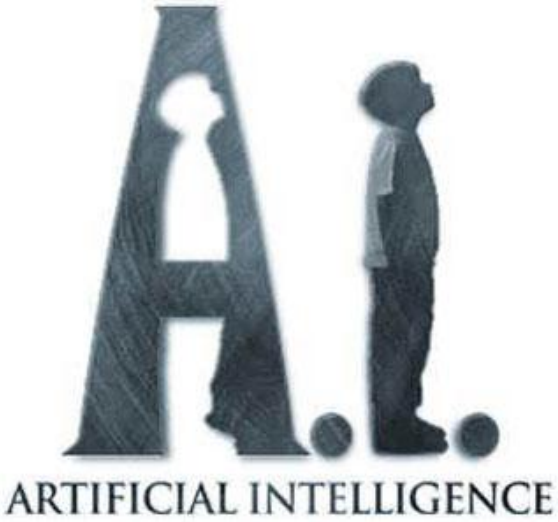
Los orígenes modernos. 1947

- William Shockley, John Bardeen y Walter Brattain inventan la "resistencia de transferencia" que mas tarde se conocerá como transistor.



Los orígenes modernos. 1956

- John McCarthy and Marvin Minsky crean, en una reunión en el Darmouth College, el concepto de Inteligencia Artificial.



Los orígenes modernos. 1957-1969

- 1957, Lenguaje FORTRAN
- 1958, Lenguaje LISP
- 1960, Lenguaje COBOL
- 1964, Douglas Engelbert introduce el concepto de Hipertexto
- 1964, Lenguaje BASIC
- 1968, HAL en 2001
- 1969, Comienzo de ARPANET

Los orígenes modernos. 1971-1973

- 1971, Intel desarrolla el 4004 el primer microprocesador.
- 1971, IBM desarrolla el primer Floppy disk
- 1973, Donald Knuth comienza a trabajar en el “Art of Programming” sentando las bases de la Ingeniería del Software.
- 1973, Robert Metcalf desarrolla en Xerox PARC (Palo Alto Research Center) el Ethernet, la forma en que se comunica una red de computadoras. El nombre hace referencia al éter, el fluido que se suponía llenaba todo el espacio; las primeras computadoras conectadas entre sí fueron bautizadas como Michelson y Morley.

Los orígenes modernos. 1974-1982

- 1974, Edward Roberts, William Yates y Jim Bybee desarrollan el Altair 8800. Tenía 256 bytes de memoria y no tenía pantalla, ni teclado ni dispositivo auxiliar de memoria.
- 1975, Bill Gates y Paul Allen escribieron el primer programa para el Altair 8800, un compilador de Basic y fundan Microsoft.
- 1977, Se desarrolla el CRAY I el primer supercomputador. Steve Jobs y Steve Wozniak desarrollan el Apple II.
- 1981, IBM crea la primera computadora personal, revolucionado la computación y su uso masivo.

Actualidad. Siglo XXI.

- Ciencia y ficción: El computador invisible (o computación ubicua). A largo plazo, el computador personal y el terminal de trabajo desaparecerán porque el acceso a la informática estará en todas partes: en las paredes, en nuestras muñecas y en “computadores para borrador” (como el papel para borrador) distribuidos para ser utilizados cuando sea necesario.

Mark Weiser, 1988

Xerox PARC

Actualidad. Siglo XXI.

- Ciencia y ficción: Efecto multiplicador de la tecnología. Si el automóvil hubiera experimentado un desarrollo parecido a la informática, se podría disponer de un Rolls-Royce por menos de $2.000. Además el vehículo dispondría de la potencia de un trasatlántico para ser capaz de recorrer un millón de kilómetros (25 veces la vuelta al mundo) con sólo un litro de gasolina.

Tom Forester, 1990

El futuro que nos alcanzó.

- La Internet.
- La inteligencia artificial (IA).
- La influencia en la educación.
- La influencia en la medicina.
- El comercio electrónico.
- La informática portátil e inalámbrica.
- El Internet de las cosas (IoT).

Historia de las Computadoras.

- Primera Generación.
  - Tubos de vacío.
  - Programadas en Lenguaje de máquina.
  - Grandes y muy costosas.
- Segunda Generación.
  - Circuitos de Transistores.
  - Programadas en lenguajes de alto nivel.

Historia de las Computadoras.

- Tercera Generación.
  - Circuitos integrados.
  - Control a través de Sistemas Operativos.
- Cuarta Generación.
  - Microprocesadores.
  - Primeros microcomputadores.
  - Aparición de ARPANET.

Historia de las Computadoras.

- Quinta Generación.
  - Microelectrónica y nanoelectrónica.
  - Competencia internacional por el dominio de la informática.
  - Internet se masifica.
  - Aparición de Computadores portátiles.
  - Manejo de Lenguaje Natural e Inteligencia Artificial.

Ambiente GNU/LINUX

Software Libre

- El término “libre” esta relacionado a la libertad y no a su costo.
- Linux es software libre, pero el software libre no es solamente GNU/Linux.
- Hay muchas licencias que son consideradas “libres”.
- Las aplicaciones propietarias o cerradas pueden trabajar conjuntamente con software libre.
- El proyecto lo empezó Linus Torvalds en 1991 y lo continúan miles de programadores en todo el mundo.

Ambiente GNU/LINUX

Beneficios del Software Libre

- Es abierto. Cualquier persona puede ver el código, modificarlo y hasta venderlo.
- Uno no se queda “pegado” con su proveedor y puede cambiar de distribución las veces que así lo desee.
- Los costos pueden reducirse ampliamente.
- Uno puede adaptar el software para que cumpla **exactamente** con los requerimientos.

Ambiente GNU/LINUX

¿Dónde puede correr GNU/Linux?

- Computadoras personales.
- Sistemas embebidos mediante la virtualización.
- Estaciones de trabajo.
- Servidores intermedios.
- Clústeres.
- Servidores de alta capacidad.

Ambiente GNU/LINUX

¿Para que puedo usar GNU/Linux?

- Servicios de red (web, correo, archivos, impresiones, seguridad, ruteo, fax, ras, irc, etc, etc, etc.)
- Renderizado de imágenes, ediciones de alta definición, etc…
- Aplicaciones de Ofimática, de edición de audio, música, video, etc…
- Ambientes de programación, big data, criptomonedas, etc…
- Etc…

Ambiente GNU/LINUX

Linux al día de hoy

- El uso de GNU/Linux se ha desarrollado ampliamente en los últimos años y sigue creciendo, por lo que es el momento de aprovechar el desarrollo que esto implica.
- La evangelización sigue siendo una tarea muy importante en el ámbito comercial e, incluso, en el académico.
- GNU/Linux no es un producto, es una herramienta para implementar una solución y eso abre un amplio campo en su uso y aprovechamiento.

Ambiente GNU/LINUX

Algunas distribuciones de Linux

- Ubuntu y sus variantes: Ubuntu GNOME (predeterminada), Lubuntu, Kubuntu, Xubuntu, Ubuntu Budgie, Ubuntu Mate, Ubuntu Kylin, Edubuntu, entre otras.
- Fedora y sus variantes: WorkStation, Server, IoT, CoreOS y SilverBlue.
- Servidores: RedHat, AlmaLinux, OpenSUSE, etc.
- Debian, LinuxMint, MX Linux, SteamOS, PureOS, etc.

Ambiente GNU/LINUX

Comandos más utilizados (Algunos recursos en línea)

- Una guía paso a paso para aprender 32 comandos Linux básicos:
  https://www.dongee.com/tutoriales/comandos-basicos-de-linux

- Los 40 comandos de Linux más utilizados que debes conocer
  https://kinsta.com/es/blog/linux-comandos

- El manual de comandos de Linux
  https://www.freecodecamp.org/espanol/news/comandos-de-linux

- Tutorial de Linux
  https://www.fing.edu.uy/inco/cursos/sistoper/recursosLaboratorio/tutorial0.pdf

Ambiente GNU/LINUX

Aplicaciones varias

- Editores (vi, emacs, nano, gedit, etc)
- Graficadores (gnuplot, kmplot, scigraphica, etc)
- Compiladores/Interpretes (Fortran, C, Python, Java, R, etc)
- Computación especializada (octave, maxima, sagemath, etc)
- Etc…

Elementos básicos de programación FORTRAN.

- Un programa de Fortran es una secuencia de líneas de texto que deben seguir una determinada sintaxis para ser un programa válido.
- Un programa de Fortran por lo general consiste de un programa principal (o main) y, posiblemente, varios subprogramas (o procedimientos o subrutinas).
- La estructura del programa principal es:

! F90/95
program *NOMBRE*
Inclusión de librerías y módulos externas
Declaraciones de variables y tipos
Instrucciones
stop
end ***program***

Notas:

(1) Las palabras que estén en *itálicas* no deberán ser tomadas en forma literal, sino como una descripción general.

(2) La sentencia **stop** es opcional y podría ser vista como redundante, pero se recomienda que el programa termine con esta sentencia para resaltar que la ejecución del programa termina ahí.

Elementos básicos de programación FORTRAN.

- Cada variable debería ser definida con una declaración, indicando el tipo de la variable. Las declaraciones más comunes son:
  - Integer :: lista de variables
  - Real :: lista de variables
  - Double precision :: lista de variables (Equivalentemente se puede usar Real*8)
  - Complex :: lista de variables
  - Logical :: lista de variables
  - Character :: lista de variables
- Aunque Fortran 77/95/2000 usa un conjunto implícito de reglas para establecer el tipo (todas las variables que comiencen con el conjunto de letras i-n son enteros, y el resto tipo real), se recomienda definir todas las variables a emplear en el programa.

Elementos básicos de programación FORTRAN.

- Una práctica recomendada es comentar las diferentes partes de un programa, ya que permite identificar lo que hace el programa, y eso contribuye a encontrar de manera más fácil posibles instrucciones erróneas.
- Los comentarios inician con el signo de admiración: “!” (sin las comillas); por lo que todo lo que se escriba a la derecha de este signo será ignorado por el compilador.
- Una segunda recomendación es usar letras minúsculas al momento de programar, aunque Fortran NO distingue entre mayúsculas y minúsculas.
- Una tercera recomendación es hacer uso de las tabulaciones para las instrucciones “anidadas”; de tal manera que se visualice con mayor claridad la jerarquía en las diferentes instrucciones.

Elementos básicos de programación FORTRAN.

Ejemplo sencillo

```
! Este programa lee un número real r
! calcula el área del círculo con radio r
! y nos muestra el resultado.
program circulo
real :: r, area
real, parameter :: pi = 3.141593
      write (*,*) 'Escribe el radio r:'
      read (*,*) r
      area = pi*r*r
      write (*,*) 'Area = ', area
stop
end
```

Elementos básicos de programación FORTRAN.

Ejemplo sencillo con doble precisión

```
! Este programa lee un número real r
! calcula el área del círculo con radio r
! y nos muestra el resultado.
program circulo
real*8 :: r, area
real*8, parameter :: pi = 4.0d0*atan(1.0d0)
      print *, 'Escribe el radio r:'
      read *, r
      area = pi*r*r
      print *, 'Area = ', area
stop
end
```

Elementos básicos de programación FORTRAN.

Ejemplo con repetición de cálculos (Usando DO - EndDo)

```
! Este programa calcula el factorial de un entero n.
program factorial
integer :: i, n
real :: factor
real*8 :: factord
      factor = 1.0e0
      factord = 1.0d0
! Aquí leemos el número al que queremos calcularle su factorial: n!
      print *, 'Dame el entero n para calcularle su factorial n!'
      read *, n
```

(continúa en la siguiente diapositiva)

Elementos básicos de programación FORTRAN.

Ejemplo con repetición de cálculos (Usando DO - EndDo)

(…. Continuación)

```
! Se inicia el calculo iterativo
      do i = 1, n
            factor = factor * i
            factord = factord * i
      end do
! Presentamos los resultados
      print *, 'El factorial de ', n, 'usando precisión sencilla es ', factor
      print *, 'El factorial de ', n, 'usando doble precisión es ', factord
stop
end
```

Solución de ecuaciones de una variable.

1. Método de bisección

2. Método de Newton(-Raphson)

3. Método de la secante

Método de bisección

El método de bisección se basa en el teorema de Cálculo conocido como "*Teorema del valor medio*".

"Sea $f(x)$ continua en un intervalo $[a, b]$ y supongamos que $f(a) < f(b)$. Entonces, para cada z tal que $f(a) < z < f(b)$, existe un x_0 en el intervalo (a, b) tal que $f(x_0) = z$".

La misma conclusión se obtiene para el caso que $f(a) > f(b)$.

Método de bisección

Básicamente, el Teorema del valor medio nos dice que toda función continua en un intervalo cerrado, una vez que alcanzó ciertos valores en los extremos del intervalo, entonces debe alcanzar todos los valores intermedios.

En particular, si $f(a)$ y $f(b)$ tienen signos opuestos, entonces un valor intermedio es precisamente $z = 0$, y por lo tanto, el Teorema del valor medio nos asegura que debe existir un valor x_0 en el intervalo (a, b) tal que $f(x_0) = 0$, es decir, debe haber *por lo menos* una raíz de $f(x)$ en el intervalo (a, b).

Método de bisección

El método de bisección para una función continua $f(x)$ sigue los siguientes pasos:

1. Encontrar valores iniciales x_a, x_b tales que $f(x_a)$ y $f(x_b)$ tienen signos opuestos, es decir, $f(x_a)f(x_b) < 0$.

2. La primera aproximación a la raíz se toma igual al punto medio entre x_a y x_b, es decir

$$x_\gamma = \frac{x_a + x_b}{2} \tag{1.1}$$

3. Evaluar $f(x_\gamma)$.

Método de bisección

Al evaluar $f(x_\gamma)$, necesariamente tenemos alguno de los siguientes casos:

I. $f(x_a)f(x_\gamma) < 0$, en este caso $f(x_a)$ y $f(x_\gamma)$ tienen signos opuestos y por lo tanto la raíz se encuentra en el intervalo $[x_a, x_\gamma]$.

II. $f(x_a)f(x_\gamma) > 0$, en este caso $f(x_a)$ y $f(x_\gamma)$ tienen el mismo signo y por lo tanto la raíz se encuentra en el intervalo $[x_\gamma, x_b]$.

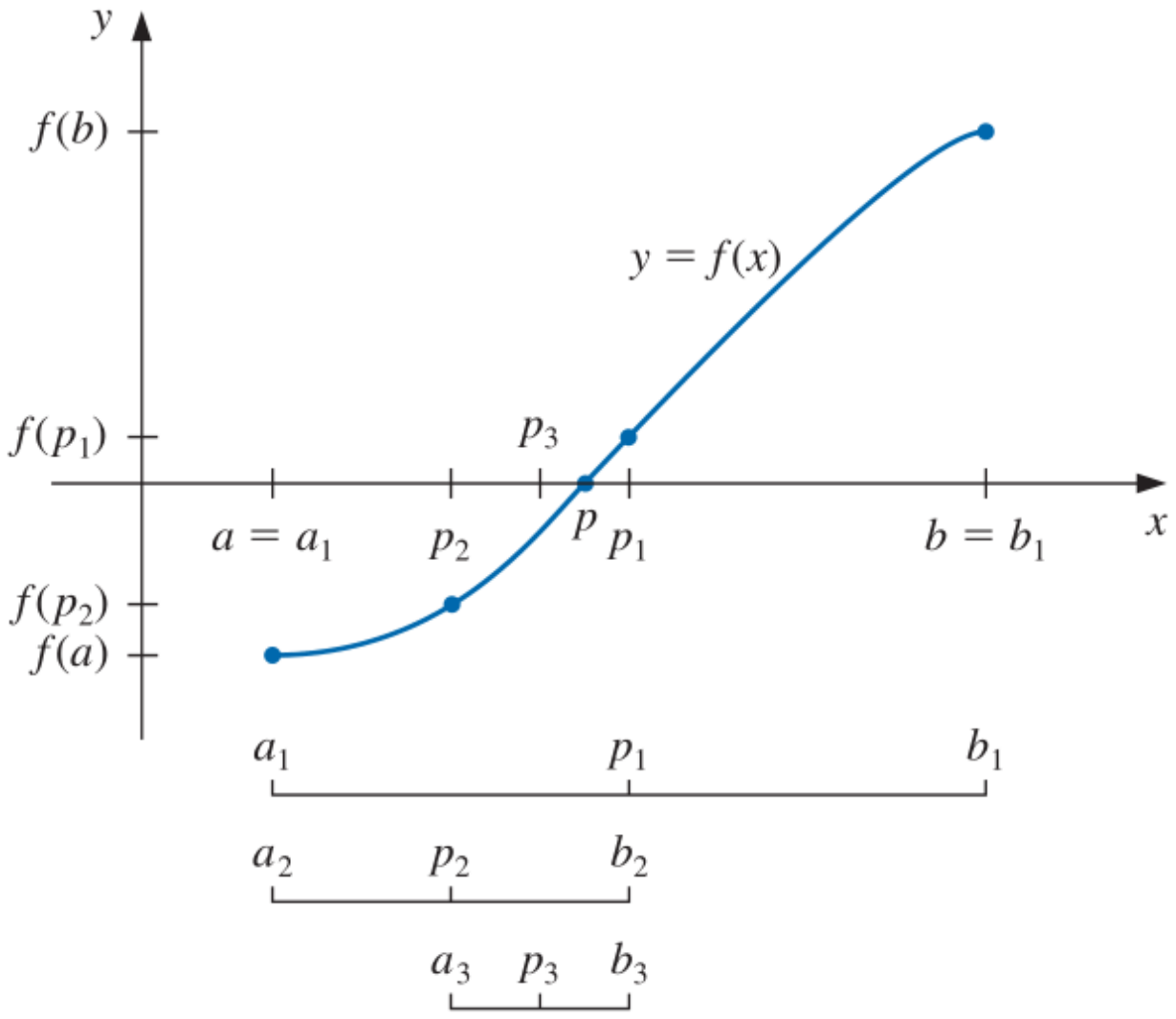
III. $f(x_a)f(x_\gamma) = 0$, en este caso se tiene que $f(x_\gamma) = 0$ y, por lo tanto, la raíz es x_γ.

Método de bisección

4. El proceso se debe repetir hasta que la diferencia entre los valores previo (x_k) y actual (x_{k+1}) de x sea menor a un cierto valor preestablecido al momento de realizar el cálculo, conocido como tolerancia ε, es decir

$$|x_{k+1} - x_k| < \varepsilon \qquad (1.2)$$

Método de bisección. Diagrama.



Método de Newton

En este método, al igual que en el de bisección, lo que buscamos es tener una forma sistemática de obtener aproximaciones a la solución de una ecuación de la forma $f(x) = 0$.

En este caso, se parte del desarrollo de Taylor para la función $f(x)$ centrada en x_0, a saber

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \tfrac{1}{2}(x - x_0)^2 f''(x_0) + \cdots = 0$$

Si asumimos que $|x - x_0|$ es muy pequeño, podemos resolver para x:

$$x \approx x_0 - \frac{f(x_0)}{f'(x_0)}$$

Método de Newton

Con la idea anterior, el método de Newton-Raphson toma como base el siguiente enunciado:

Sea f una función derivable y sea r un cero real de f. Si x_n es una aproximación a r, entonces la siguiente aproximación x_{n+1} está dada por

$$x_{n+1} \approx x_n - \frac{f(x_n)}{f'(x_n)} \tag{1.3}$$

con $f'(x_n) \neq 0$.

Método de Newton

El procedimiento que debemos seguir para emplear este método consiste en lo siguiente:

1. Observando la gráfica de la función f se estima un valor adecuado para primera aproximación x_1.

2. Sustituyendo esta primera aproximación en la ecuación (1.3) dada anteriormente, se obtiene una segunda aproximación x_2 dada por

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} \qquad (1.4)$$

Método de Newton

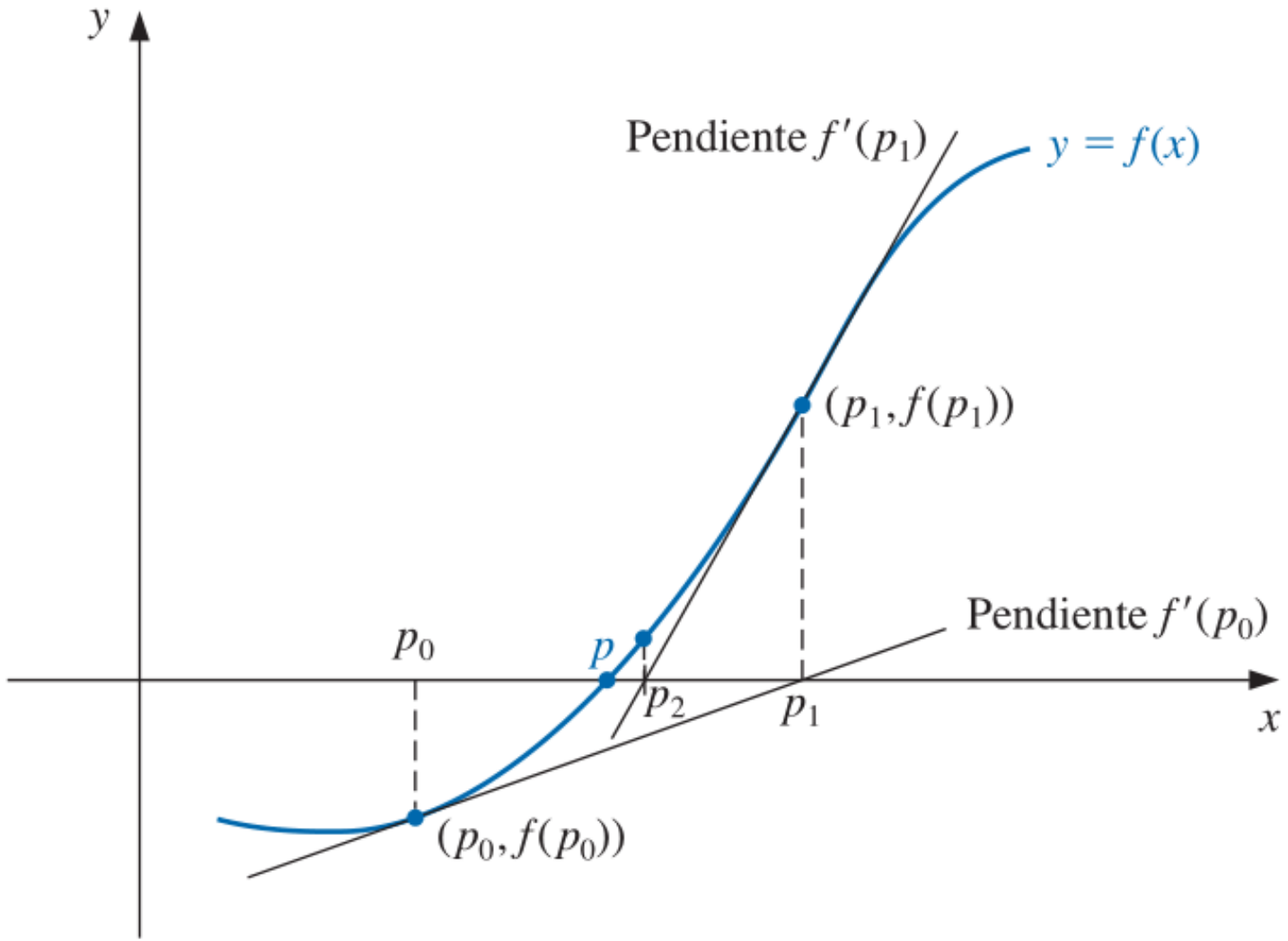
El procedimiento que debemos seguir para emplear este método consiste en lo siguiente:

3. Luego, se calcula x_3 sustituyendo en la ecuación (1.3) la segunda aproximación x_2; y así sucesivamente hasta que se cumple la siguiente expresión:

$$|x_{k+1} - x_k| < \varepsilon \tag{1.5}$$

donde ε se conoce como la tolerancia y se establece antes de iniciar el cálculo de la raíz.

Método de Newton. Diagrama.



Método de la secante

Este método se basa en el método de Newton, pero evita el cálculo de la derivada usando la siguiente aproximación:

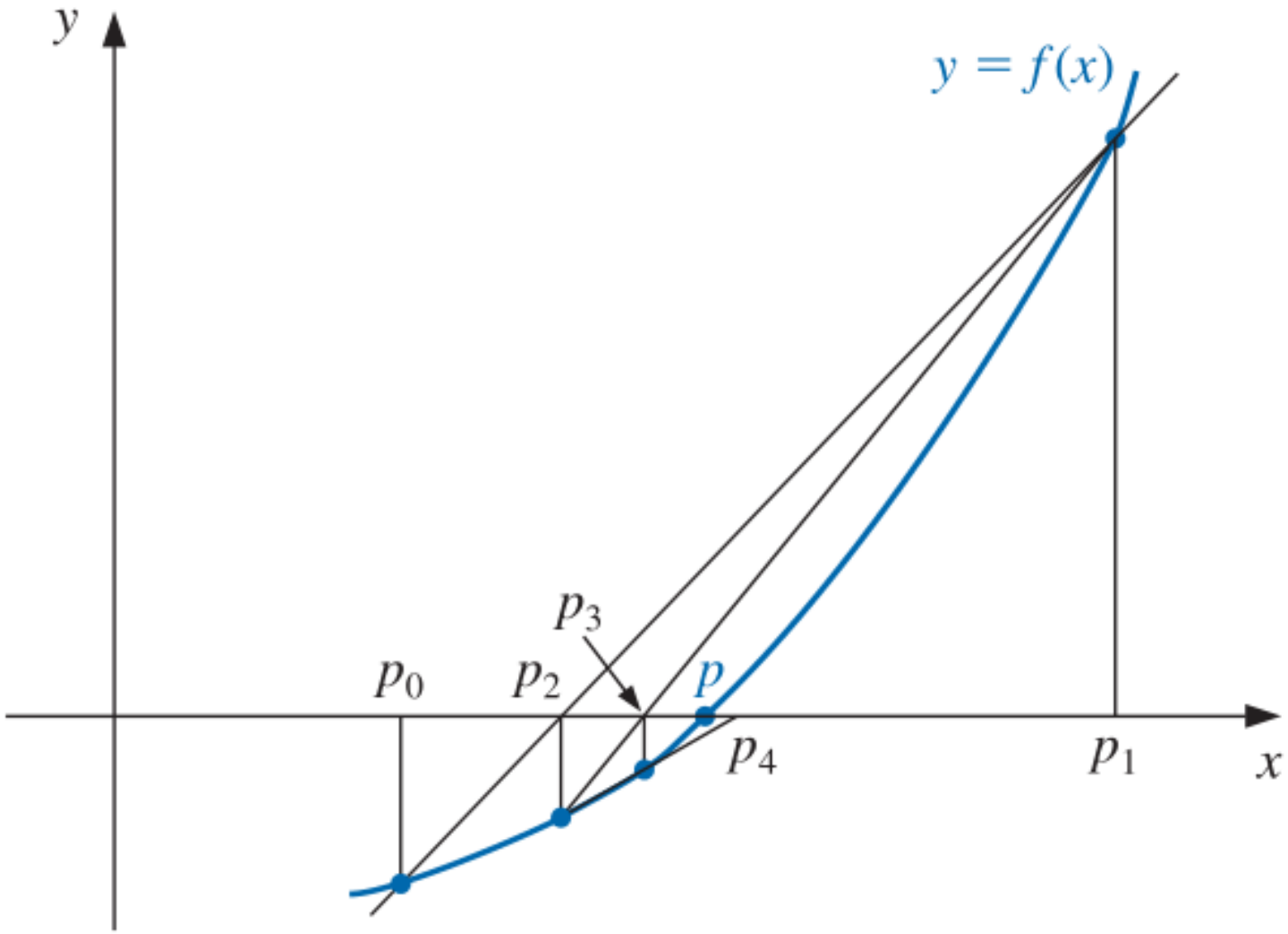
$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} \tag{1.6}$$

que al sustituir en la fórmula de Newton, nos permite obtener

$$x_{n+1} \approx x_n - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})} \tag{1.7}$$

Es importante mencionar que este método requiere dos aproximaciones a la raíz, a diferencia del método de Newton que sólo requiere uno.

Método de la secante. Diagrama



Física Computacional I

Dr. Roberto Pedro Duarte Zamorano

Universidad de Sonora