FACULTAD DE INGENIERIA
Universidad de Buenos Aires

# Trabajo Práctico 1

[75.12] Análisis Numérico I
Segundo cuatrimestre de 2018


| Alumno | Padrón | Mail |
|---|---|---|
| del Mazo, Federico | 100029 | delmazofederico@gmail.com |
| Kristal, Juan Ignacio | 99779 | kristaljuanignacio@gmail.com |


Curso 07:

- Dr Daniel Fabian Rodriguez
- Valeria Machiunas
- Federico Balzarotti
- Michael Portocarrero

**ANÁLISIS NUMÉRICO I - 75.12 – 95.04**

**Curso: Rodríguez- Balzarotti - Machiunas- Portocarrero**

**2º cuatrimestre de 2018**

# TRABAJO PRÁCTICO DE MÁQUINA Nº 1

**Desarrollo del práctico:**

1) Programar un algoritmo para estimar la unidad de máquina ($\mu$), en simple y en doble precisión.
2) Implementar el método de trapecios compuestos para evaluar la integral:

a) $$F(\alpha,\beta)=\int_{1}^{240}\frac{\sin(Px)+\beta x^2}{\alpha\ x}dx$$

donde P = (N° de padrón de integrante 1 + N° de padrón de integrante 2) /50

o bien P = Nº de padrón / 25

$\alpha$=0.17 y $\beta$=0.41

de tal forma que el módulo del error absoluto de truncamiento sea menor que $10^{-5}$.

Informar qué valor de n (cantidad de trapecios) se ha utilizado, justificando la elección.

Considere P exacto, y $\alpha$ y $\beta$ bien redondeados.

3) Fijado dicho valor de n, luego:
   I. Calcular la condición del problema mediante la técnica de perturbaciones experimentales.
   II. Estimar experimentalmente el término de estabilidad.
   III. Utilizando los resultados anteriores que sean necesarios, y suponiendo nulo el error inherente, acotar el error total.
   IV. Repetir III suponiendo que el error inherente relativo está acotado por $0.5\ 10^{-4}$
   V. Indicar la fuente más importante de error en los dos casos anteriores.

**La entrega del presente trabajo práctico deberá realizarse de acuerdo al reglamento del curso, en la fecha informada en clase.**

# Índice

# 1. Introducción

El trabajo práctico tiene como objetivo el cálculo y acotamiento de errores de la siguiente integral:

$$F(\alpha, \beta) = \int_1^{240} \frac{\sin(Px) + \beta x^2}{\alpha x} dx$$

Siendo:

- $P = \frac{\sum_{padrones}}{50} = \frac{99779+100029}{50} = 3996{,}16$ exacto
- $\alpha = 0{,}17$ bien redondeando
- $\beta = 0{,}41$ bien redondeando

Específicamente:

- Se estimará el valor de la unidad de maquina $\mu$.
- Se evaluará la integral con el método de trapecios compuestos teniendo con objetivo en mente que el error absoluto de truncamiento sea menor a $1 \times 10^{-5}$.
- Se calculará computacionalmente la cantidad de trapecios utilizada en el método descrito anteriormente.
- Se calculará la condición del problema mediante perturbaciones experimentales.
- Se estimará experimentalmente el término de estabilidad.
- Se acotará el error total.

# 2. Desarrollo

## 2.1. Estimación de $\mu$

Para los cálculos del $\mu$ se utilizó el algoritmo del ejemplo 6.4 del libro de Hernan Gonzales [1].

## 2.2. Función y sus derivadas

Siempre teniendo en cuenta los valores de $P, \alpha, \beta$ previamente utilizados, definimos la funcion $f(x)$ como:

$$f(x) = \frac{\sin(Px) + \beta x^2}{\alpha x} dx$$

Graficamos la función para saber un poco más de ella en la figura 1

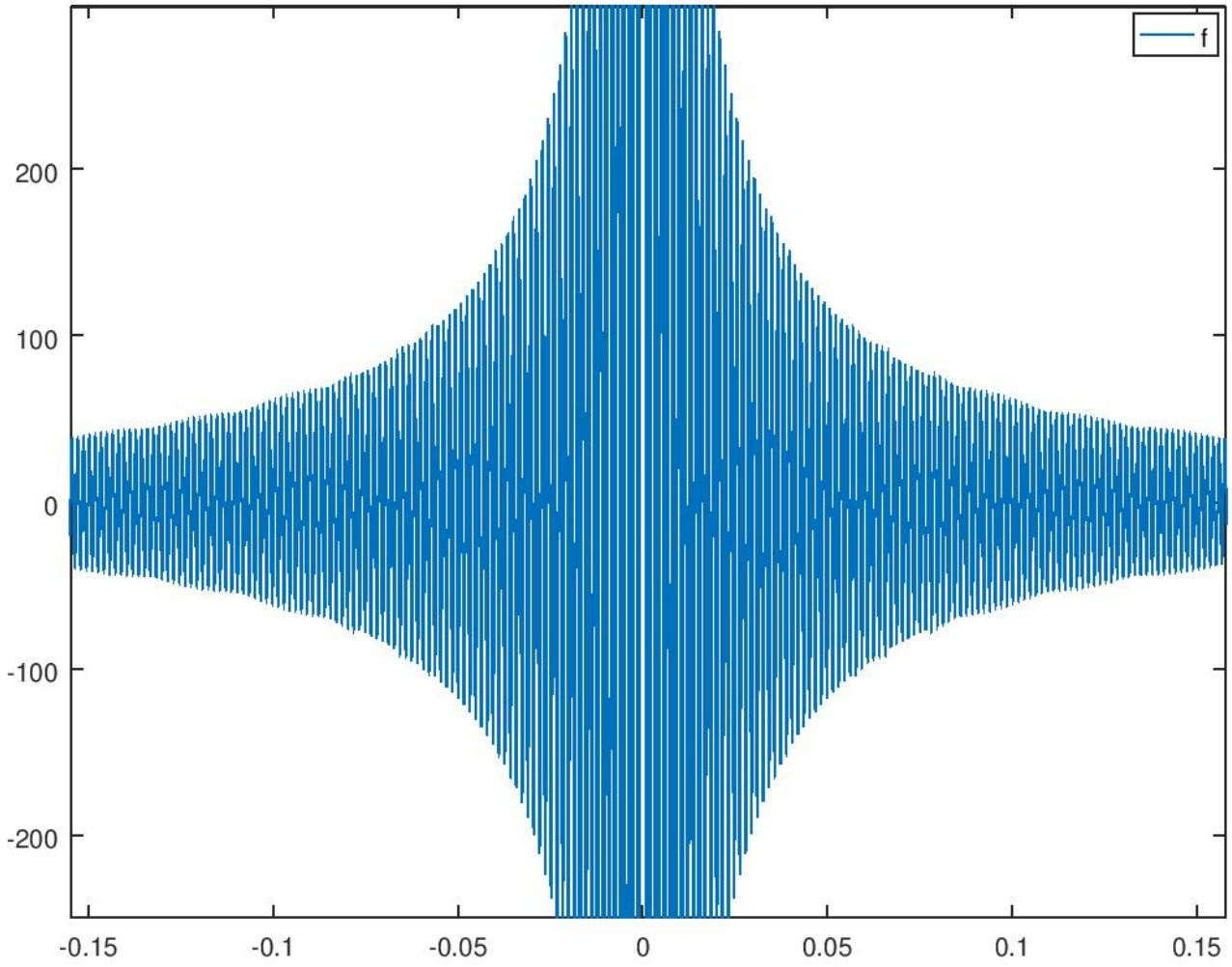


Figura 1: $f(x)$

De esta función calculamos sus derivadas y las graficamos en las figuras 2 y 3, para utilizar en cálculos posteriores.

$$f'(x) = \frac{P\cos(Px)}{\alpha x} - \frac{\sin(Px)}{\alpha x^2} + \frac{\beta}{\alpha}$$

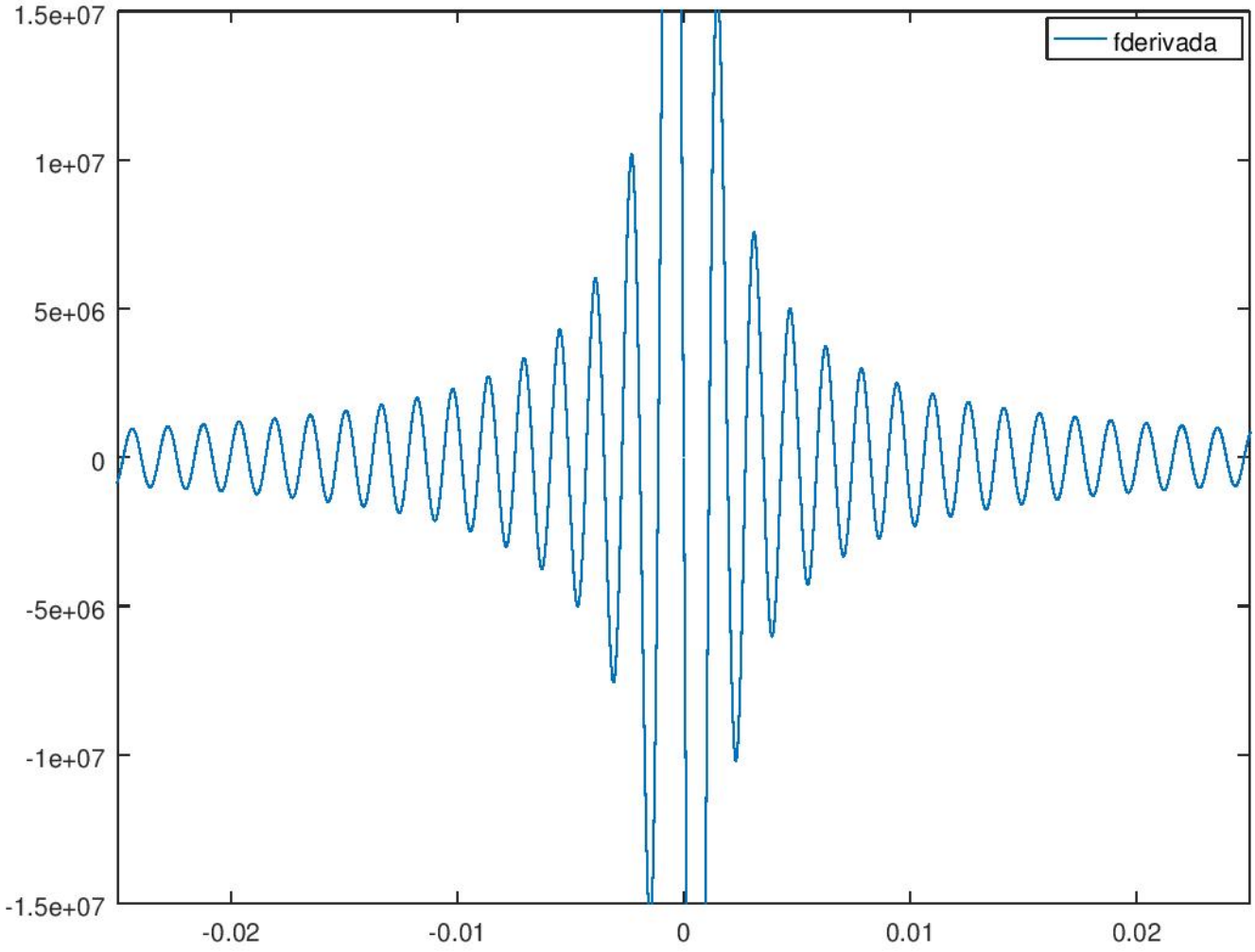


Figura 2: $f'(x)$

$$f''(x) = -\frac{2P\cos(Px)}{\alpha x^2} + \frac{2\sin(Px)}{\alpha x^3} - \frac{P^2\sin(Px)}{\alpha x}$$

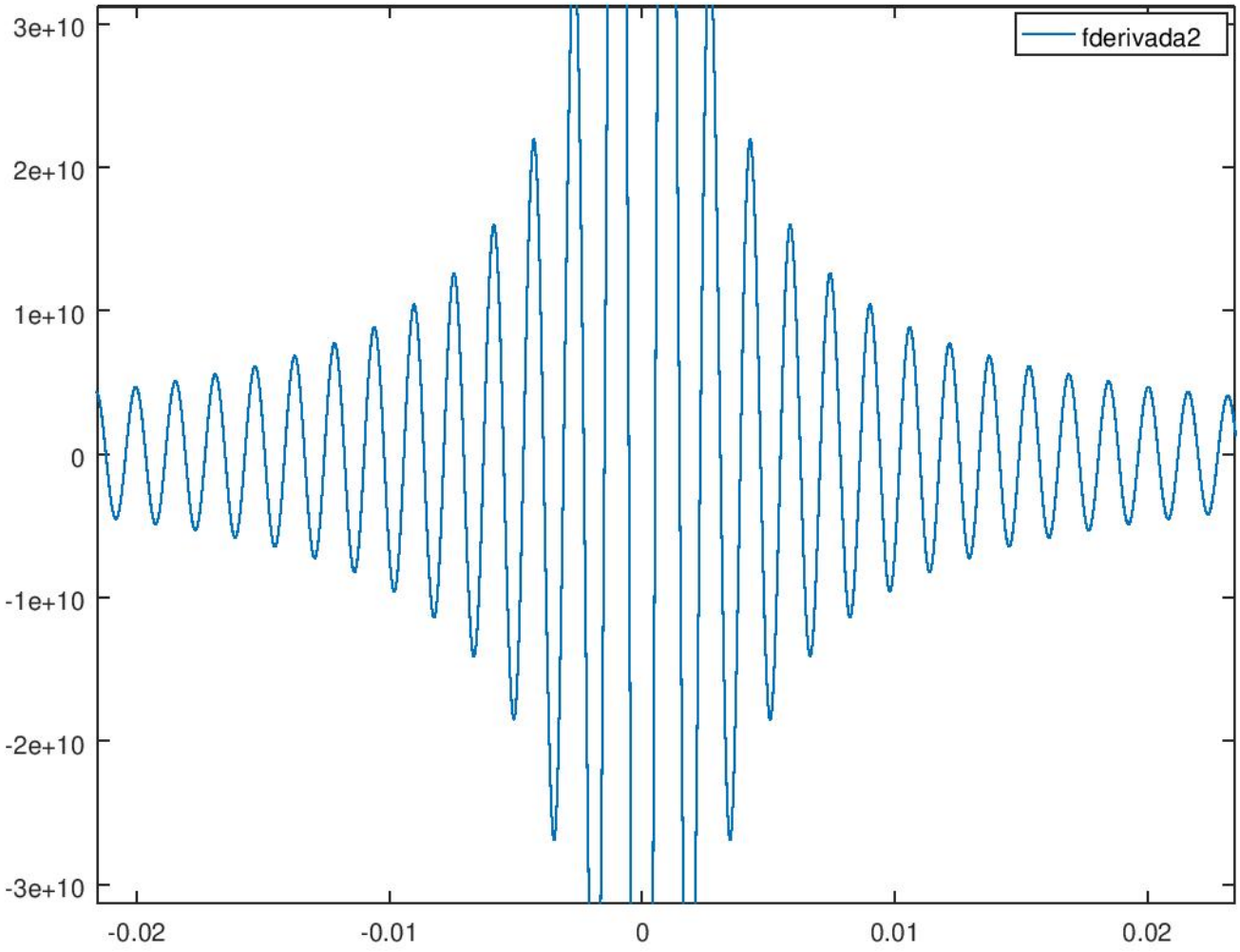


Figura 3: $f''(x)$

## 2.3. Método de trapecios compuestos

Sabemos que el error de truncamiento producido por el método de trapecios compuestos es:

$$\epsilon_t = -\frac{(b-a)^3}{12n^2} * f''(\xi)$$

Donde $b, a$ son los limites de integración y $n$ es la cantidad de trapecios. Como lo que queremos es acotar el error de truncamiento, debemos evaluar a la segunda derivada en su imagen máxima, es decir $\xi = 1$

Por lo tanto, y ahora con el error de truncamiento al que queremos llegar, despejamos la cantidad de trapecios:

$$n = \sqrt{\left| -\frac{(b-a)^3 * f''(1)}{12\epsilon_t} \right|}$$

Es con este n que se puede finalmente implementar la función de el método de los trapecios compuestos.

### 2.3.1. Truncamiento de n

Al despejar por el método de los trapecios compuestos el n necesario para tener el error deseado se notó que este valor era de tal magnitud y orden que computacionalmente carecería de sentido usarlo para cada cálculo. Es por esto que se decidió hacer un truncamiento de este valor, para poder tratarlo como es debido y en un lógico margen de tiempo. De todas formas, solo anecdóticamente, se incluye una corrida del programa con el n original.

El criterio para truncar n es el de ver como escala el cálculo de la integral respecto del valor, y luego decidir un punto de corte tratable arbitrariamente (en nuestro caso, 5 minutos). Se puede ver en el gráfico 4 que esta es una función lineal, lo cual tiene sentido ya que lo único que adiciona computacionalmente es el ciclo definido `for` de la función, haciendolo $\mathcal{O}(n)$, siendo n el mismo n con el que venimos tratando, redundantemente.

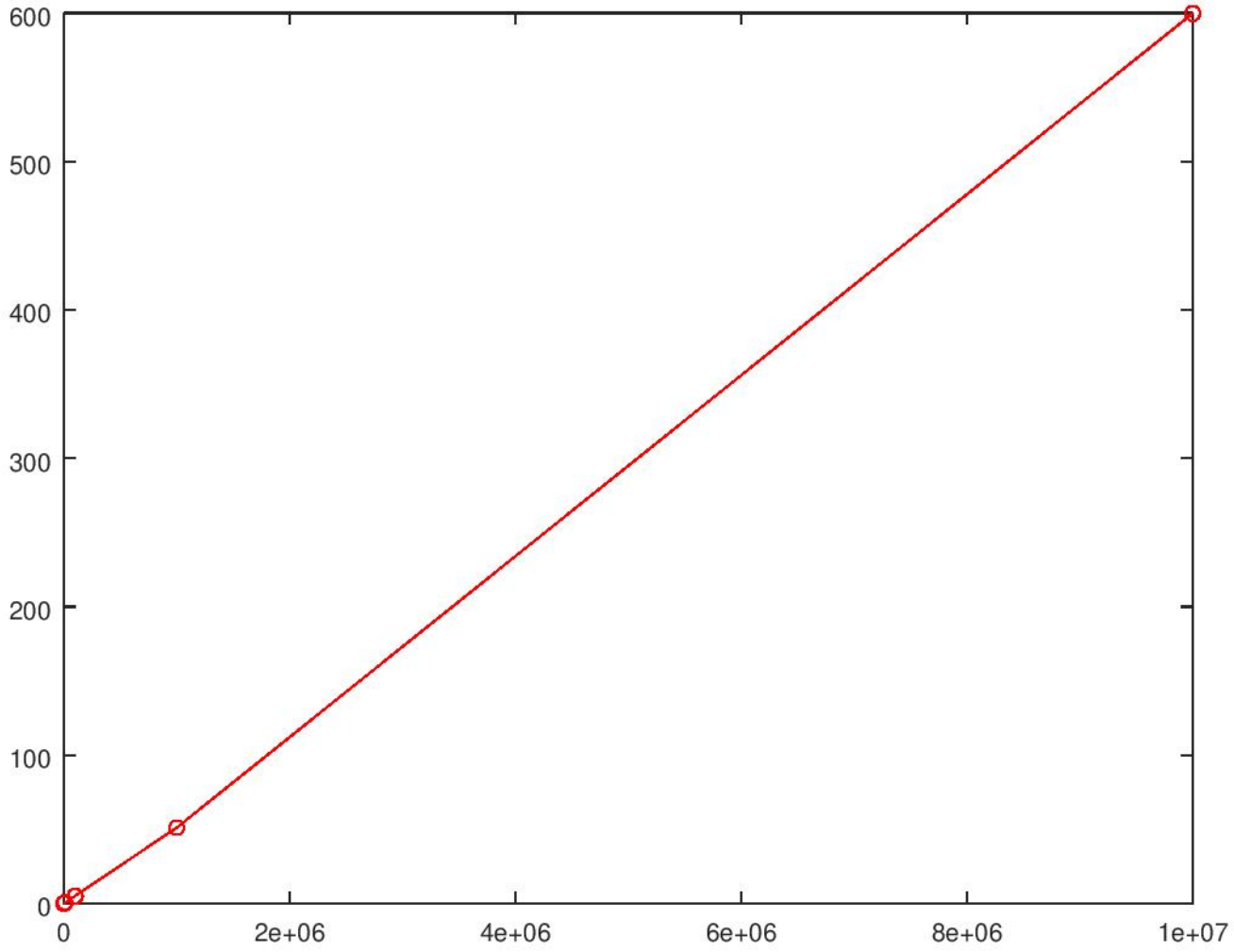


Figura 4: Calculo de la integral para distintos n en funcion del tiempo

## 2.4. Condición del problema y término de estabilidad

# 3. Resultados

## 3.1. Estimación de $\mu$

Con el algoritmo utilizado se llego al resultado $1 \times 10^{-8}$ para el $\mu$ de precisión simple y $1 \times 10^{-16}$ para el $\mu$ de precisión doble.

## 3.2. Método de trapecios compuestos

### 3.2.1. Precisión simple

Haciendo los calculos se vió que n es igual a $2.4160 \times 10^8$, y luego se decidió truncar el número n a $5000000$ para que sea un cálculo tratable. Para el n original la integral dió como resultado $6.9458 \times 10^4$, mientras que para n truncado dió

### 3.2.2. Precisión doble

Haciendo los calculos se vió que n es igual a $241403216$, y luego se decidió truncar el número n a $5000000$. Para el n original la integral dió como resultado $6.9458 \times 10^4$, mientras que para n truncado dió

# 4. Conclusiones

# A. Anexo I: Código Fuente

mu.m

```
function mu
  mu_simple
  mu_doble
end

function mu_doble
  mu_doble=1; digitos=1; x=2;
  while (x>1)
    digitos = digitos+1;
    mu_doble = mu_doble/10;
    x = 1+mu_doble;
  endwhile
  mu_doble
end

function mu_simple
  mu_simple=single(1); digitos=single(1); x=single(2);
  while (x>1)
    digitos = digitos+1;
    mu_simple = mu_simple/10;
    x = 1+mu_simple;
  endwhile
  mu_simple
end
```

main.m

```
function integral = main
  padron1 = 100029; padron2 = 99779;
  global P = (padron1 + padron2) / 50
  global ALPHA = 0.17; global BETA = 0.41;
  global ERR_MAX = 10e-5;
  global LIM_INF = 1; global LIM_SUP = 240;
  global MUS = 10e-8;
  #n = calcular_n(ERR_MAX)
  n = 500;
  integral_d = 6.9458e+04;
  integral_s = 7.0236e+04;
  te = abs(calcular_te(integral_d, integral_s))
  err = te.*MUS;
  cpa = [];
  cpb = [];
  integral = calcular_area(n)
  for i = 1:16;
    perturbacion = 1/(10 .^i);
    cpa = [cpa, perturbarA(perturbacion,n)];
    cpb = [cpb, perturbarB(perturbacion,n)];
   end
   cpa_max = max(cpa)
   cpb_max = max(cpb)
end

function te = calcular_te(d,s)
  global MUS
  te = (d.-s)./(d.*(MUS));
end

function cp = perturbarA(perturbacion,n)
  global ALPHA
  ALPHA += perturbacion;
  valor_perturbado_sup = calcular_area(n);
  ALPHA -= 2 .*perturbacion;
  valor_perturbado_inf = calcular_area(n);
  ALPHA += perturbacion;
  cp = abs((1 .- (valor_perturbado_inf ./ valor_perturbado_sup)) ./ perturbacion);
end

function cp = perturbarB(perturbacion,n)
  global BETA
```

```
  BETA += perturbacion;
  valor_perturbado_sup = calcular_area(n);
  BETA -= 2 .*perturbacion;
  valor_perturbado_inf = calcular_area(n);
  BETA += perturbacion;
  cp = abs((1 .- (valor_perturbado_inf ./ valor_perturbado_sup)) ./ perturbacion);
end

function y = f(x)
  global P ALPHA BETA

  y = ( sin(x.*P) + BETA * (x.^2) ) ./ (x.*ALPHA);
end

function y = fderivada(x)
  global P ALPHA BETA

  primer_term = (P./(ALPHA.*x)) .* cos(P.*x);
  segundo_term = - ( ( sin(P.*x) ) ./ ( ALPHA .* (x.^2) ) );
  tercer_term = BETA / ALPHA;
  y = abs(primer_term) + abs(segundo_term) + abs(tercer_term);
end

function y = fderivada2(x)
  global P ALPHA BETA

  primer_term = - (2.*P.*cos(P.*x) ) ./ (ALPHA .* (x.^2) );
  segundo_term = 2* sin(P.*x) ./ (ALPHA .* (x.^3) );
  tercer_term = - ( ( (P^2)*sin(P.*x) ) ./ (ALPHA .* x) );
  y = abs(primer_term) + abs(segundo_term) + abs(tercer_term);
end

function n = calcular_n(error_maximo_truncamiento)
  global LIM_SUP LIM_INF
  num = - ( (LIM_SUP - LIM_INF)^3 ) * fderivada2(1);
  denom = error_maximo_truncamiento * 12;
  n = sqrt(abs(num/denom));
end

function a = calcular_area(n)
  global LIM_SUP LIM_INF;
  h = ( LIM_SUP - LIM_INF ) ./ n;
  f_inicio = f(LIM_INF) / 2;
```

```
  f_fin = f(LIM_SUP) / 2;
  f_i = 0;
  for i = 1:n-1;
    f_i = f_i + f(LIM_INF + i*h) * h;
  end
    a = ( f_inicio + f_fin ) * h + f_i;
end



function y = derivada_alpha()
  global ALPHA BETA P LIM_INF LIM_SUP;
  y = - 1/(ALPHA^2) * integral(( sin(x.*P) + BETA * (x.^2) ) ./ (x.*ALPHA),LIM_INF,
      ↪ LIM_SUP);
end

function y = derivada_blpha()
  global ALPHA BETA LIM_INF LIM_SUP;
  y = integral(x./ALPHA,1,240);
end
```

singlemain.m

```
function integral = singlemain
  padron1 = single(100029); padron2 = single(99779);
  global P = (padron1 + padron2) / single(50)
  global ALPHA = single(0.17); global BETA = single(0.41);
  global ERR_MAX = single(10e-5);
  global LIM_INF = single(1); global LIM_SUP = single(240);
  #m = calcular_n(ERR_MAX)
  tic
  n = single(5000000)
  integral = calcular_area(n)
  toc
end

function y = f(x)
  x = single(x);
  global P ALPHA BETA

  y = ( single(sin(x.*P)) + BETA * (x.^2) ) ./ (x.*ALPHA);
end

function y = fderivada(x)
  x = single(x);
  global P ALPHA BETA

  primer_term = (P./(ALPHA.*x)) .* single(cos(P.*x));
  segundo_term = - ( ( single(sin(P.*x)) ) ./ ( ALPHA .* (x.^2) ) );
  tercer_term = BETA / ALPHA;
  y = abs(single(primer_term)) + abs(single(segundo_term)) + abs(single(tercer_term)
      ↪ );
end

function y = fderivada2(x)
  x = single(x);
  global P ALPHA BETA

  primer_term = - (2.*P.*single(cos(P.*x)) ) ./ (ALPHA .* (x.^2) );
  segundo_term = 2* single(sin(P.*x)) ./ (ALPHA .* (x.^3) );
  tercer_term = - ( ( (P^2)*single(sin(P.*x)) ) ./ (ALPHA .* x) );
  y = abs(single(primer_term)) + abs(single(segundo_term)) + abs(single(tercer_term)
      ↪ );
end
```

```
function n = calcular_n(error_maximo_truncamiento)
  global LIM_SUP LIM_INF
  num = - ( (LIM_SUP - LIM_INF)^3 ) * fderivada2(1);
  denom = error_maximo_truncamiento * 12;
  n = single(sqrt(abs(num/denom)));
end

function a = calcular_area(n)
  n = single(n);
  global LIM_SUP LIM_INF;
  h = ( LIM_SUP - LIM_INF ) ./ n;
  f_inicio = f(LIM_INF) / 2;
  f_fin = f(LIM_SUP) / 2;
  f_i = 0;
  for i = 1:n-1;
    f_i = f_i + f(LIM_INF + i*h) * h;
  end
    a = ( f_inicio + f_fin ) * h + f_i;
end
```

Generación de graficos

```
fplot(@f, [-0.02 0.02])
fplot(@fderivada, [-0.02 0.02])
fplot(@fderivada2, [-0.02 0.02])
```

Truncamiento y gráfico de n

```
function y = graficar_n()
x = [1,10,100,1000,10000,100000,1000000,10000000];
y = [];
for n = x
    n
    tic;
    integral = calcular_area(n)
    y = [y,toc];
    printf("Tiempo␣=␣%ds\n\n",y(length(y)))
end
plot(x,y,'o-r')
end
```

## B. Anexo II: Resultados Numéricos

Resultados de `mu.m`

```
>> mu
mu_simple = 1.0000e-08
mu_doble = 1.0000e-16
```

Resultados originales sin truncamiento de n

```
>> main
n = 2.4160e+08
Elapsed time is 12404.8 seconds.
ans = 6.9458e+04
```

n con precisión simple

```
>> main
n = 241403216
```

Resultados con truncamiento de n

```
>> main
n = 5000000
integral = 6.9458e+04
Elapsed time is 360.336 seconds.
>> singlemain
n = 5000000
integral = 7.0236e+04
Elapsed time is 450.23 seconds.
```

Cálculos hechos para el criterio de truncamiento de n

```
>> graficar_n
n = 1
integral = 6.9497e+04
Tiempo = 0.000295877s

n = 10
integral = 6.9459e+04
Tiempo = 0.000695944s

n = 100
integral = 6.9465e+04
Tiempo = 0.00530505s

n = 1000
```

```
integral = 6.9465e+04
Tiempo = 0.0516629s

n = 10000
integral = 6.9458e+04
Tiempo = 0.509583s

n = 100000
integral = 6.9458e+04
Tiempo = 5.11305s

n = 1000000
integral = 6.9458e+04
Tiempo = 51.1415s

n = 10000000
integral = 6.9458e+04
Tiempo = 599.773s
```

## Bibliografía

[1] Gonzales, Hernan: *Análisis Numérico, Primer Curso* Buenos Aires: Nueva Librería, 2002.