

Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Hồ Chí Minh
Khoa Điện tử Viễn thông



BÁO CÁO ĐỒ ÁN CUỐI KÌ

MÔN: Thực hành MATLAB

Họ và tên: Nguyễn Minh Tâm

MSSV: 20200333

Giảng viên phụ trách môn: Thái Hồng Hải

Ca học : 01

Đồ án: 3

Nội dung đồ án: Thiết kế giao diện triển khai 3 phương pháp tính gần đúng tích phân (hình thang, Simpson và chính xác). Giao diện gồm các Edit Field (Text) để nhập hàm $f(x)$, khoảng tính tích phân $[a;b]$, số đoạn con N . Sử dụng duy nhất 1 Button để:

- Tính ra giá trị tích phân bằng cả 3 phương pháp
- Hiện thị kết quả bằng Label theo định dạng có 10 chữ số thập phân sau dấu phẩy.
- Vẽ đồ thị $f(x)$ trong khoảng tính tích phân, sử dụng hàm `linspace` để chia đều các điểm.

MỤC LỤC

1. Giới thiệu đồ án:	2
2. Tổng quan đồ án:	2
3. Phần code đồ án:	4
3.1. Code thuật toán	4
3.1.a/ Phương pháp tính gần đúng tích phân hình thang	4
3.1.b/ Phương pháp tính gần đúng tích phân Simpson	6
3.1.c/ Tích phân chính xác	7
3.2. Code giao diện	8
3.2.a/ Code giao diện các hàm bắt lỗi mà MATLAB không tự bắt lỗi được	8
3.2.b/ Code giao diện hiển thị kết quả	19
3.2.c/ Phần code bắt các lỗi mà MATLAB có thể bắt được	25
4. Tổng kết đồ án	30

1. Giới thiệu đồ án:

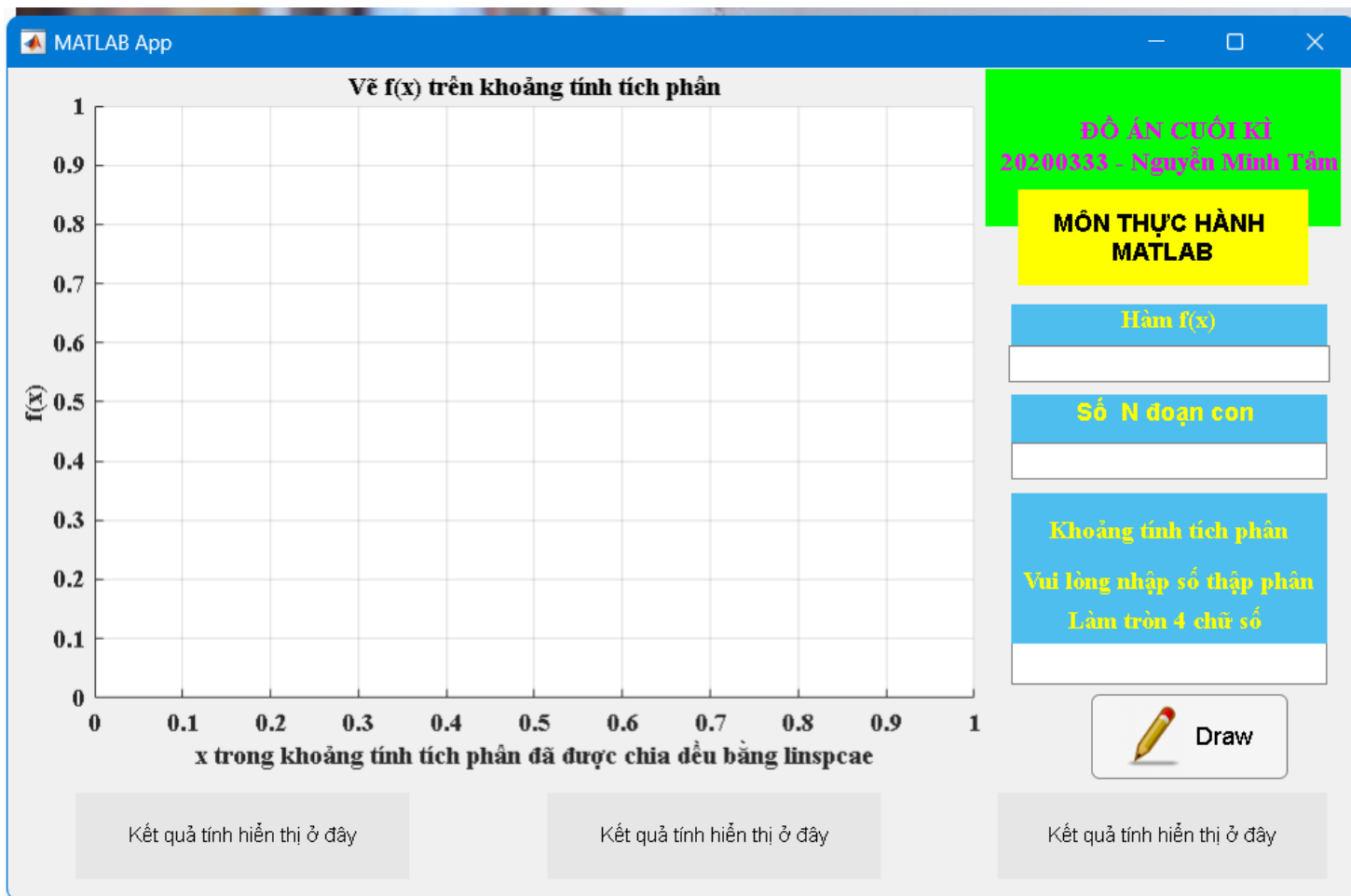
Về đồ án thì em áp dụng theo các kiến thức đã học ở các bài thực hành đó là Lab 05, Lab 03 và Lab 01 để áp dụng vào trong việc thực hiện các code thuật toán, code giao diện và thiết kế giao diện tổng quan cho đồ án của em – là giao diện triển khai 3 phương pháp tính gần đúng tích phân. Đồ án sau khi em xây dựng sẽ đảm bảo sự thân thiện với người sử dụng – người tương tác, đảm bảo về mặt thẩm mỹ và sẽ dễ dàng sử dụng hơn. Ở môn lý thuyết em học phương pháp tích phân Simpson $\frac{1}{3}$ và tích phân Simpson $\frac{3}{8}$ nhưng ở môn thực hành này em được thầy giới thiệu theo công thức tích phân Simpson $\frac{1}{3}$ nên em xin phép sử dụng công thức trên để tính toán. Em đã thiết kế ứng dụng kiểm soát lỗi do người dùng nhập sai định dạng tốt, hàm tính của em đạt giá trị tin cậy cao, đã áp dụng các kiến thức của bài học trên lớp và xây dựng nên ứng dụng giao diện tính tích phân bằng ba cách và các yêu cầu phía sau của đồ án

2. Tổng quan đồ án:

- Về đồ án em sẽ có một nút nhấn duy nhất vì theo yêu cầu đề là sử dụng duy nhất 1 button để cho app của em có thể tính toán và thực thi yêu cầu đề. Với nút nhấn thì em đã thêm icon cho ứng dụng của mình thêm phần sinh động, bắt mắt. Đối với nút nhấn thì em sẽ lập trình hàm lắng nghe và bắt sự kiện để thực thi các lệnh em đã lập trình ở hàm bắt sự kiện đó. Trước hết thì nói về các phương pháp tính tích phân, theo yêu cầu đề là viết từng function riêng nên em cũng đã viết ra từng function riêng trong hàm bắt sự kiện nút nhấn. Các function của 3 phương pháp tính em sẽ đề đầu tiên, sau đó là tới các hàm kiểm tra điều kiện gồm hàm bắt dữ liệu, dữ liệu đầu tiên hàm sẽ lấy và kiểm tra là hàm $f(x)$ sau đó sẽ đến số đoạn con và cuối cùng là khoảng phân li nghiệm.
 - Để nói về việc kiểm tra hàm $f(x)$ được nhập thì app của em sẽ kiểm tra xem có đúng định dạng hay không, có đủ ẩn hàm hay không và kiểm tra xem đã nhập hay chưa để báo lỗi. Còn nếu không có lỗi sẽ đọc vào data cho code xử lý. Chi tiết về đoạn code kiểm tra này và giải thích sẽ được đề cập ở phần 3 của bài báo cáo này.
 - Để nói về kiểm tra số đoạn con thì đơn giản hơn là sẽ kiểm tra xem số đoạn con đã nhập hay chưa và số N đoạn con đã nhập có dư biến hay không để báo lỗi. Nếu thỏa hết điều kiện thì sẽ chuyển sang bước tiếp theo là kiểm tra khoảng tính tích phân. Chi tiết về đoạn code và giải thích code sẽ được em đề cập đến ở phần 3 của bài báo cáo này
 - Cuối cùng sẽ đến với kiểm tra là khoảng tính tích phân $[a; b]$ có thỏa điều kiện hay chưa. Ở đây yêu cầu khoảng tính tích phân được nhập vào có định dạng là $[a;b]$ nên em sẽ tiến hành cho MATLAB kiểm tra xem định dạng nhập vào có đúng định dạng đó hay không, nếu thiếu dấu ngoặc vuông, hoặc giữa a và b không có dấu chấm phẩy, số a bằng b thì đều sẽ báo lỗi và không thực hiện chương trình. Chi tiết đoạn code và phần giải thích code sẽ được em đề cập ở phần 3 của bài báo cáo này
 - Khi một trong các lỗi được phát hiện và thông báo thông qua dòng pop – up ra màn hình thì ứng dụng của em sẽ ngưng hết việc tính toán, reset dữ liệu và reset cả đồ thị, để người dùng thấy và tiến hành nhập lại, đến khi không có lỗi nào bị phát hiện

thì sẽ có một dấu hiệu cho chương trình thực thi diễn ra. Chi tiết về hàm kiểm tra này cũng sẽ được em đề cập và giải thích chi tiết ở phần 3 của bài báo cáo này.

- Ngoài các lỗi em đã đề cập ở trên thì trong quá trình thực hiện app và chạy kiểm thử sẽ dẫn đến lỗi khác, thì em cũng đã lập trình cho app để có thể xuất ra màn hình dòng báo lỗi và mô tả lỗi chi tiết để ta biết lỗi ở đâu nhằm khắc phục. Chi tiết ở phần 3 bài báo cáo này
- Sau phần code kiểm tra điều kiện thì em sẽ tiến hành đến phần gọi function và thực hiện việc lập trình cho app của mình có thể xuất kết quả ra Label của mình. Để sắp xếp các ô Label nhằm hiển thị kết quả theo yêu cầu đề là 10 chữ số thập phân thì em đã tô xám nền của mỗi ô, với việc tô như vậy sẽ giúp em nhận biết là các ô có đều nhau hay chưa, có ô nào gần hơn ô kia hay có ô nào lại ngắn hơn ô kia hay không, và việc xuất kết quả ra label sẽ được thực hiện bằng hàm Text đã được matlab hỗ trợ. Chi tiết sẽ được đề cập kĩ hơn ở phần 3 bài báo cáo này
- Ngoài ra theo yêu cầu đề là vẽ đồ thị $f(x)$ trong khoảng tính tích phân thì em cũng đã tiến hành thêm giao diện đồ thị, vẽ thêm các ô lưới cho chúng và thêm tiêu đề cho đúng yêu cầu. Trang trí bằng cách tô màu, tô nền và làm chữ đậm giúp đồ án của em không bị nhàm chán mà thêm phần sinh động và đẹp mắt.
- Sau đây là giao diện tổng quan về ứng dụng mà em đã thiết kế với đồ án



3. Phần code đồ án:

Ở phần trình bày code này thì em xin phép sẽ trình bày theo phong cách là chia phần code của em gồm 2 phần chính đó là code thuật toán và code giao diện. Với mỗi phần code của mình thì em sẽ trình bày code trước sau đó là phần giải thích chi tiết đoạn code của mình. Đối với code thuật toán thì em xin phép trình bày theo cách là ghi ra cách tính tích phân xong tới code và cuối cùng là giải thích và sẽ đến phương pháp tiếp theo cũng tương tự là như vậy. Ví dụ đối với phương pháp tích phân hình thang thì em sẽ ghi phương pháp, code và giải thích code sau đó đến phương pháp tích phân Simpson cũng sẽ là code và giải thích code sau đó đến phương pháp chính xác cũng là code và giải thích code. Còn đối với code giao diện thì em chia ra ba phần là code kiểm tra nếu app không tự bắt lỗi được, thứ hai là code hiển thị kết quả và cuối cùng là code phần MATLAB tự bắt lỗi được. Trong phần giải thích em sẽ có kèm theo hình ảnh minh họa về sự giải thích đó để trực quan phần giải thích của em. Sau đây là phần code của đồ án của em.

3.1. Code thuật toán

3.1.a/ Phương pháp tính gần đúng tích phân hình thang

➤ *Code:*

```
function y = tichphanhinhthang(f,a,b,N)

    h = (b-a) / N;

    y = 0;

    temp = (f(a) + f(b)) * (h/2);

    for i = 1:N-1

        z = a + i*h;

        y = f(z) + y;

    end

    y = h*y + temp;

end
```

➤ *Giải thích code:*

Đầu tiên thì em đến với function của tính gần đúng tích phân hình thang với input vào là hàm f, a, b và số đoạn con N. Với a và b là khoảng tính tích phân, f là hàm f(x) đã được nhập từ UI. Và output sẽ được lưu ở biến y. Tiếp theo thì đến đoạn code tính toán, ta có công thức tính gần đúng tích phân hình thang:

$$I = \int_a^b f(x)dx \approx \frac{h}{2} [f(a) + f(b) + 2 \sum_{i=1}^{N-1} f(a + ih)], \text{ bước nhảy } h \text{ sẽ là:}$$

$h = \frac{b-a}{N}$. Từ các công thức đó em có đoạn code đầu tiên để tính được bước nhảy: $h = (b-a) / N$. Sau khi tính được h , em cho $y=0$ để tính tổng bằng phương pháp cộng dồn. Có y và h ban đầu, em tiến hành tính $(f(a) + f(b)) \times \frac{h}{2}$ thông qua lệnh $\text{temp} = (f(a) + f(b)) * (h/2)$, biến temp sẽ lưu kết quả hiện tại và em còn cần tính $\frac{h}{2} \times (2 \times \sum_{i=1}^{N-1} f(a + ih))$ nên tiếp theo em có vòng for để thực hiện tính tổng $\sum_{i=1}^{N-1} f(a + ih)$. Đó là lý do em có dòng for $i = 1:N-1$ và code trong đó là em sẽ có biến tạm z để thực hiện tính $a + i*h$ sau đó sẽ thế z vào f để tính $f(z) \xrightarrow[\text{tương đương}]{} f(a + ih)$. Sau đó sẽ tính tổng bằng lệnh $y = f(z) + y$; sau vòng for đó ta đã tính ra được $\sum_{i=1}^{N-1} f(a + ih)$ đang được lưu là biến y . Sau khi thoát khỏi vòng for thì ta tiến hành cộng các kết quả với $(f(a) + f(b)) \times \frac{h}{2}$ đang được lưu là temp , còn $\sum_{i=1}^{N-1} f(a + ih)$ đang được lưu là biến y còn $\frac{h}{2} \times 2 \times \sum_{i=1}^{N-1} f(a + ih)$ thì rút gọn thành $\frac{h}{2} \times 2 \times \sum_{i=1}^{N-1} f(a + ih)$ và từ tất cả dữ liệu đó em có y kết quả là: $\text{temp} + h*y$, nên em có lệnh $y = h*y + \text{temp}$. Sau đó thu được output y . Đó là phần thuật toán và function tính toán. Sau khi có được function trên, thì ở các hàm về sau chỉ cần gọi đến function và truyền hàm f_x , số a , số b và số N thì sẽ thu được output y là kết quả tính gần đúng tích phân hình thang. Sau đây sẽ là code của phương pháp tính gần đúng tích phân Simpson

3.1.b/ Phương pháp tính gần đúng tích phân Simpson

➤ *Code:*

```
function y_simpson = tichphanSimpson(f,a,b,N)

h = (b - a)/N;

temp = (h/3)*(f(a)+f(b));

y_simpson = 0;

y1 = 0;

for i = 2: N-1

    check = mod(i,2);

    if check == 0

        z = a + i*h;

        f(z);

        y_simpson = y_simpson + f(z);

    elseif check ~= 0

        z = a + i*h;

        f(z);

        y1 = y1 + f(z);

    end

end

y_simpson = (y_simpson*2 + y1*4)*h/3 + temp;

end
```

➤ *Giải thích code:*

Tiếp theo thì em đến với function của tính gần đúng tích phân Simpson với input vào là hàm f, a, b và số đoạn con N. Với a và b là khoảng tính tích phân, f là hàm f(x) đã được nhập từ UI. Và output sẽ được lưu ở biến y_simpson. Theo tích phân bằng phương pháp Simpson thì đầu tiên ta cần tìm h với $h = \frac{b-a}{N}$, sau khi có h thì ta tiến hành tính $\frac{h}{3}(f(a) + f(b))$ vì theo công thức của tính gần đúng tích phân với công thức Simpson là: $y = \frac{h}{3} \left[f(a) + f(b) + 4 \sum_{i=1}^{N-1} f(a + i \cdot h) \right]$

$ih) + 2 \sum_{\substack{i=2 \\ i \text{ chẵn}}}^{N-1} f(a + ih) \Bigg] \text{ thì em tính } \frac{h}{3}(f(a) + f(b)) \text{ và lưu vào biến temp}$
 thông qua lệnh $\text{temp} = (h/3)*(f(a)+f(b))$. Tiếp theo thì em sẽ cho $y1 = 0$ và $y = 0$ tức là em sẽ tính tổng chẵn trước và tổng chẵn sẽ được lưu ở tên y , tức là ở đây em tính đầu tiên là $\sum_{\substack{i=2 \\ i \text{ chẵn}}}^{N-1} f(a + ih)$ như ở cách tính của function tính gần đúng tích phân hình thang thì em dùng biến tạm z để tính ra $a + i*h$ và $f(z)$ sẽ tương đương với $f(a+ih)$ của công thức, $y = 0$ để cộng dồn nhằm tính ra tổng bằng cách cộng dồn nên đối với $\sum_{\substack{i=2 \\ i \text{ chẵn}}}^{N-1} f(a + ih)$ sẽ được đặt trong vòng for với i bắt đầu từ 2 và sẽ chạy đến $N-1$, mỗi lần chạy sẽ thực thi lệnh chia lấy dư khi lấy $i \bmod 2$ và lưu vào biến check , sau khi thực hiện lệnh chia lấy dư thì em kiểm tra nếu $\text{check} == 0$ tức là số chẵn thì sẽ thực thi lệnh ở trong còn nếu khác 0 thì bỏ qua và cộng i cho 1 và cứ thế đến khi hoàn thành ở $N-1$ thì ngưng. Sau cùng, sau vòng for đầu thì $\sum_{\substack{i=2 \\ i \text{ chẵn}}}^{N-1} f(a + ih)$ đã được lưu vào y thông qua lệnh $y = y + f(z)$. Sau khi tính được $\sum_{\substack{i=2 \\ i \text{ chẵn}}}^{N-1} f(a + ih)$ thì còn $\sum_{\substack{i=1 \\ i \text{ lẻ}}}^{N-1} f(a + ih)$ nên em tiến hành chạy vòng for tiếp theo để tính. Đầu tiên em cho vòng for chạy từ 1 đến $N-1$, với mỗi lần chạy thì hàm sẽ tính chia i lấy dư cho 2 và lưu vào biến check , sau đó hàm sẽ kiểm tra nếu biến check mà khác 0 – tức i là lẻ thì thực hiện lệnh trong vòng if, ở đây cách tính ở trong giống với khi xét i chẵn nên giống cách tính trên nên em xin phép không giải thích vì nó giống ở trên, chỉ khác là thay vì y thì sẽ là $y1$ để là biến lưu sau khi tính tổng với $y1 = y1 + f(z)$. Sau khi thực hiện tính $\sum_{\substack{i=1 \\ i \text{ lẻ}}}^{N-1} f(a + ih)$ và $\sum_{\substack{i=2 \\ i \text{ chẵn}}}^{N-1} f(a + ih)$ thì em thu được hai biến $y1$ và y , sau đó em sẽ lấy $y1*2$, $y_simpson*4$ và hai cái cộng nhau để cho giống công thức là $4 \sum_{\substack{i=1 \\ i \text{ lẻ}}}^{N-1} f(a + ih) + 2 \sum_{\substack{i=2 \\ i \text{ chẵn}}}^{N-1} f(a + ih)$, sau khi cộng thì nhân với $(\frac{h}{3})$ và lưu vào biến y luôn để tiết kiệm bộ nhớ và tiết kiệm nhiều biến nhất có thể. Như vậy ta đã có đầy đủ và từ giải thích trên em có lệnh $y_simpson = (y_simpson * 2 + y1 * 4) * h/3 + \text{temp}$. Và có được $y_simpson$ – là output cần có được. Từ function này thì các phần code phía sau thì chỉ cần gọi lại function thì ta thu được kết quả.

3.1.c/ Tích phân chính xác

➤ Code:

```
function y_chinhxac = tichphanChinhxac(f,a,b)

    y_chinhxac = integral(f,a,b);

end
```

➤ Giải thích code:

Tiếp theo thì đây là function của tính chính xác tích phân bằng lệnh integral. Do ở đây hàm f đã truyền vào có kiểu dữ liệu là function_handles, nên kiểu dữ liệu của hàm f(x) là function – đã đúng theo yêu cầu của lệnh integral nên em không cần chuyển sang kiểu dữ liệu function nữa. Các input sẽ là a,b,f còn output là y_chinhxac. Và khi ta gọi function là tíchphanChinhxac thì sẽ thu được kết quả tính chính xác tích phân.

3.2. Code giao diện

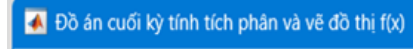
3.2.a/ Code giao diện các hàm bắt lỗi mà MATLAB không tự bắt lỗi được

- Ở đây để đồng bộ về duy nhất 1 cách nhập để MATLAB có thể hiểu và lấy được dữ liệu thì em sẽ lập trình để MATLAB chỉ đọc khi người dùng nhập vào khoảng tính tích phân [a;b] thỏa điều kiện là $a \neq b$; N có duy nhất 1 số và N sẽ lớn hơn 0 và cấu trúc yêu cầu khi nhập phải là [a;b]. Các trường hợp không thỏa đều sẽ báo lỗi và chương trình sẽ không thực thi lệnh. Đoạn code sẽ là:
 - *Code bắt lỗi nhập không đúng định dạng là phải có dấu ngoặc vuông*



```
app.UIFigure.Name = 'Đồ án cuối kỳ tính tích phân và vẽ đồ thị f(x)';
try
    check_1 = 0;
    check_temp = 0;
    check_temp1 = 0;
    check_temp2 = 0;
    check_temp3 = 0;
    fx = app.HmfxEditField.Value;
    fx = str2func(['@(x)', fx]);
    data = app.KhongtnhtchphnEditField.Value;
    l = length(data);
    if ((data(1) ~= '[') || (data(l) ~= ']'))
        msg = {'Lỗi xác định khoảng tính tích phân, Vui lòng nhập lại theo đúng cú pháp "[a;b]" '};
        msgbox(msg, 'Error', 'error');
    else check_temp = 1;
end
```

- *Giải thích code:*

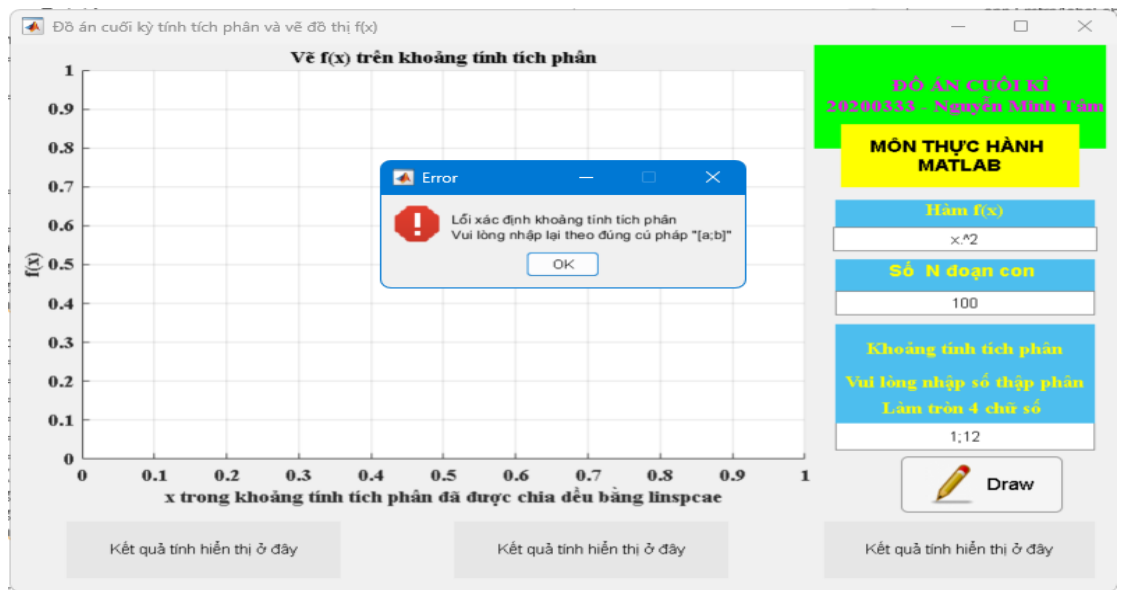
Đầu tiên thì em sẽ cho mỗi khi nút DRAW được nhấn thì phần hiển thị tên app trên tool bar sẽ hiển thị ra là: “Đồ án cuối kỳ tính tích phân và vẽ đồ thị $f(x)$, kết quả của dòng code đó sẽ là hiển thị :



Để bắt đầu chương trình thì em có cấu trúc try catch. Thì cấu trúc lệnh này dùng để thực thi các lệnh ở giữa chúng. Tức là bất cứ lệnh nào nằm giữa try và catch thì sẽ đều thực thi, tuy nhiên nếu trong quá trình thực thi lệnh mà phát sinh lỗi, trả ra lỗi thì lệnh sẽ ngưng và trả về lỗi. Đối với các lỗi thì nếu ta không lập trình hàm xử lý lỗi sau catch để thực hiện thì matlab sẽ ngưng toàn bộ chương trình chính, trả kết quả lỗi thông qua việc trở đến dòng code để báo vàng. Do code của em dài nên để dễ trình bày thì em đã cắt từng phần chức năng riêng biệt để dễ quan sát và để em giải thích code nên ở đây em để try mà không thấy catch, nhưng ở phần sau em sẽ giải thích kĩ về catch và các lỗi mà MATLAB tự bắt được. Đó là lý do có try ở ngay đầu. Tiếp theo em sẽ có biến check_temp là 0 tức ban đầu em sẽ kiểm tra xem từng điều kiện có bị sai hay không, nếu bị sai thì biến check của em sẽ là 0 và chỉ khi nào tất cả biến check_temp gồm check_temp, check_temp1, check_temp2 và check_temp3 là 1 tức là thỏa điều kiện thì em mới thực thi lệnh và ra kết quả. Có 4 điều kiện em sẽ kiểm tra, mà một trong đó là điều kiện hiện em đang giải thích tức hàm nhập phải có định dạng có ngoặc vuông mở và đóng. Biến trả kết quả kiểm tra sẽ là check_temp, với mặc định là 0, chỉ khi không thỏa điều kiện sai thì sẽ được gán cho giá trị 1. Trước khi vào hàm kiểm tra điều kiện đầu tiên thì em sẽ lấy dữ liệu từ ô data nhập vào để nếu thiếu hàm nào thì MATLAB sẽ có lỗi khi bắt dữ liệu sẽ xuất ra màn hình. Đầu tiên là hàm nhận giá trị hàm $f(x)$ với lệnh `fx = app.HmfxEditField.Value`, với ô để nhập hàm $f(x)$ sẽ có tên là `app.HmfxEditField`, và để lấy giá trị ra thì ta cần sử dụng hàm `Value` có trong thư viện lệnh của app designer matlab. Dữ liệu sau khi lấy từ ô `app.HmfxEditField` sẽ có dữ liệu là string, mà để MATLAB có thể xử lý và xuất dữ liệu thì ta cần hàm $f(x)$ phải là biến có kiểu dữ liệu func nên có lệnh `str2func`, đó là lý do em có lệnh `fx = str2func(['@(x)', fx])`. Tiếp theo thì em sẽ kiểm tra điều kiện của khoảng tính tích phân, để kiểm tra thì ta cần lấy giá trị từ ô đã nhập ra sau đó mới có thể kiểm tra được. Dữ liệu được lấy từ ô lấy giá trị tính tích phân chúng được nhập từ ô `app.KhongtnhtchphnEditField`, do lấy giá trị ra nên có lệnh `.Value`, sau khi lấy ta sẽ có dữ liệu là kiểu string, nên em có lệnh: `data = app.KhongtnhtchphnEditField.Value`, biến data đang lưu dữ liệu mà người dùng nhập từ UI. Tiếp theo thì em cần lấy độ dài của mảng data, data của em sẽ được nhập lần lượt theo ô `app.KhongtnhtchphnEditField` đã đọc, tức là từng phần tử, từng ký tự đã được đọc vô từng vị trí trong mảng data đó,

để làm rõ về vấn đề này em sẽ có ví dụ là nếu em có nhập phần tử vào ô app.KhongtnhtchphnEditField là [1;20] thì em sẽ có data(1) là ký tự '[', data(2) là ký tự '1', data 3 là ';', data(4) là ký tự '2', data(5) là ký tự '0' và ở data(6) sẽ là ']'. Vậy nên em cần xác định vị trí đầu và vị trí cuối để có thể kiểm tra xem dữ liệu đã nhập có đúng định dạng là "[a;b]" hay chưa. Vị trí đầu tiên thì sẽ là 1, còn vị trí cuối cùng đó sẽ là độ dài của mảng đó nên em có lệnh `l = length(data)`. Sau khi giải thích và làm rõ về những code trên, tiếp theo là hàm kiểm tra dấu ngoặc vuông là lệnh: `if ((data(1) ~= '[') || (data(l) ~= ']'))`, tức là với chỉ cần một điều kiện thỏa là hàm nhập vào không phải có định dạng "[a;b]" thì chương trình sẽ thực thi gồm các lệnh: `msg = {'Lỗi xác định khoảng tính tích phân','Vui lòng nhập lại theo đúng cú pháp "[a;b]" '}` và `msgbox(msg,'Error','error')`. Giải thích đoạn code thì trước tiên thì ta có: `msg = {'Lỗi xác định khoảng tính tích phân','Vui lòng nhập lại theo đúng cú pháp "[a;b]" '}`, sau lệnh này thì em sẽ có biến có tên là msg chứa dữ liệu là mảng ô với hai dữ liệu gồm vị trí thứ nhất – `msg{1}` là: Lỗi xác định khoảng tính tích phân và sau dấu phẩy sẽ là vị trí thứ hai – `msg{2}` là: Vui lòng nhập lại theo đúng cú pháp "[a;b]". Sau lệnh trên thì em sẽ có một mảng dữ liệu gồm các dòng thông báo lỗi. Tiếp theo là sau khi có mảng dữ liệu gồm các dòng thông báo thì em sẽ tiến hành thông báo pop – up tức là tạo hộp thoại tin nhắn. Giải thích chi tiết thì có đầu tiên là lệnh `msgbox` sẽ là lệnh dùng để tạo hộp thoại tin nhắn, chỉ số ở trong thì đầu tiên `msg` là biến đang được sử dụng để lưu dữ liệu dưới dạng mảng ô, thì em dùng cách này để có thể làm ngắn gọn hơn cho dòng code `msgbox`, vì `msg` gồm có 2 phần tử ở 2 vị trí, khi em gọi `msg` thì sẽ lần lượt truyền để xuất ra dòng hộp thoại tin nhắn đầu tiên là `msg{1}` là “Lỗi xác định khoảng tính tích phân” và `msg{2}` là “Vui lòng nhập lại theo đúng cú pháp "[a;b]"” và khi truyền theo thứ tự đó thì sẽ xuất ra hộp thoại tin nhắn dòng trên là của `msg{1}` và dòng thứ hai là `msg{2}`. Tiếp theo thì em sẽ có dòng hiển thị trên hộp thoại tin nhắn là Error có dạng là , tiếp theo thì em sẽ có dòng icon cảnh báo lỗi ở dòng hộp thoại tin nhắn thoại qua lệnh `error` là: .

Sau khi thực hiện các lệnh trên thì nếu khoảng tính tích phân mà không đúng theo cấu trúc là có dấu ngoặc vuông ở đầu và dấu ngoặc vuông ở cuối thì sẽ có dòng thông báo pop – up là:



Trong trường hợp mà nếu không có nhập đúng theo cú pháp là có dấu ngoặc vuông ở hai đầu thì lệnh if sẽ không thỏa điều kiện và sẽ rẽ nhánh tới câu lệnh else, và thực hiện cho biến check_temp lên là 1, như vậy nhập đúng cú pháp thì thông qua đoạn lệnh này sẽ thu được check_temp thỏa đầu tiên. Vẫn còn 3 điều kiện còn lại gồm không có dấu “;” ở giữa a và b, kiểm tra đảm bảo a khác b và N không âm và chỉ là duy nhất 1 số . Cả 3 điều kiện còn lại sẽ xét và giải thích ở đoạn sau

➤ *Code bắt lỗi nhập a và b mà ở giữa không có dấu “;”*

```
if (temp_1 == pi)
    t = 2;
elseif temp_1 == -pi
    t = 3;
elseif (temp_1/pi) == round((temp_1/pi),4) && (temp_1 ~=0)
    t = numel(num2str(temp_1/pi));
    t = t + 3;
else t = numel(num2str(temp_1));
end

t = t + 2;

if ((data(t) ~= ";"))
    msg = {'Lỗi xác định khoảng tính tích phân','Vui lòng nhập lại theo đúng cú pháp "[a;b]" '};
    msgbox(msg,'Error','error');
else check_temp1 = 1;
end
```

➤ *Giải thích code*

Tiếp theo thì theo yêu cầu đề là cách thức nhập khoảng tính tích phân là: $[a; b]$, ở trên em đã kiểm tra là định dạng nhập cần có dạng có dấu mở ngoặc vuông và đóng ngoặc vuông thì ở bước tiếp theo em sẽ cần kiểm tra xem giữa số a và b có dấu “;” hay không thì em sẽ thực hiện việc kiểm tra đó. Dữ liệu kiểu string sau khi thay đổi từ string sang số ví dụ như chuỗi “a[1,2]” thì khi chuyển sang số sẽ là mảng $a = 1 \ 2$, hay ví dụ như chuỗi “a[1;2]” thì khi chuyển sang số sẽ là mảng $a = \frac{1}{2}$. Nên không còn dấu hay ký tự khác số nữa mà sẽ có hoàn toàn là kiểu dữ liệu số nên để thực hiện việc kiểm tra là đúng định dạng là $[a;b]$ hay chưa thì phải thực hiện ngay ở dữ liệu string. Ý tưởng em đặt ra là đầu tiên em sẽ chuyển từ dữ liệu string sang số và lưu vào biến tạm; sau đó em sẽ tìm độ dài của số ở vị trí đầu tiên tức là em sẽ bỏ qua vị trí đầu tiên, vị trí của các số đang được xem là ký tự ở vị trí trước dấu chấm phẩy để kiểm tra tại vị trí đặt dấu chấm phẩy. Khi có được độ dài của số ở vị trí trước dấu chấm phẩy thì em sẽ cộng 1 ở vị trí đầu và cộng

thêm 1 tại vị trí cần kiểm tra thì sẽ ra vị trí chính xác của dấu chấm phẩy cần kiểm tra trong mảng data được lấy từ app.KhongtnhtchphnEditField. Đó là ý tưởng về cách kiểm tra xem khoảng tính tích phân có dấu “;” ở giữa a và b hay không. Sau đây là giải thích về đoạn code em đã trình bày ở trên. Đầu tiên em sẽ đặt biến tạm là a_1 sẽ lưu kết quả của việc chuyển từ kiểu dữ liệu string sang số của biến data, nên em có lệnh: `a_1 = str2num(data)`. Tiếp theo là em lấy phần tử đầu tiên của mảng a_1, vì theo ý tưởng của em thì em sẽ tìm độ dài của số ở vị trí đầu nên em sẽ có lệnh: `temp_1 = a_1(1)`, sau lệnh này thì em sẽ có biến temp_1 sẽ lưu số ở vị trí đầu tiên của mảng a_1 sau khi biến đổi từ string sang số. Tiếp theo em có biến t là biến tạm dùng để lưu số, ở các dòng code sau em sẽ trình bày chi tiết ở sau về lý do có biến t này, biến tạm này em đặt bằng 0 như là phần bắt đầu. Tiếp theo đó thì em sẽ kiểm tra xem temp_1 có bằng pi hay không vì khi chuyển từ string sang số thì sẽ là một dãy số và khi ta thấy số chữ số của pi thì sẽ là một số có rất nhiều chữ số nên nếu xét số chữ số của pi thì độ dài của pi sẽ lớn hơn 2 ký tự “pi”, nên pi là số đặc biệt và lý do em xét riêng ở đây. Các trường hợp em xét ở đây là pi, -pi, n*pi và -n*pi. Đầu tiên thì kiểm tra xem temp_1 có phải là pi hay không thông qua `if (temp_1 == pi)` nếu là pi thì ta thấy pi sẽ có 2 ký tự nên sẽ có `t = 2`. Tiếp theo thì nếu temp_1 không phải là pi thì ta kiểm tra xem temp_1 có bằng -pi không vì thông thường nếu nhập thì người dùng sẽ nhập “-pi” chứ không ai nhập “-1*pi” nên ở đây em xét hai trường hợp riêng là $\pm pi$. Ta thấy “-pi” sẽ có 3 ký tự nên em sẽ có `t = 3`. Tiếp theo thì hai trường hợp còn lại là “n*pi” và “-n*pi”. Đối với hai trường hợp này thì em sẽ xét chung trong một lệnh `if`. Ở giao diện người dùng em cũng đã giới hạn là chỉ nhập số thập phân đã làm tròn 4 chữ số nên ở đây việc kiểm tra sẽ dễ dàng hơn. Giải thích thì đầu tiên em sẽ kiểm tra xem có phải dạng n*pi hay là chỉ là số n bình thường thông qua lệnh: `(temp_1/pi) == round((temp_1/pi),4)`, tức là em sẽ tách n ra khỏi pi để thực hiện việc xét xem có phải n*pi hay không, nếu đúng thì sau khi chia pi sẽ còn n và việc làm tròn sẽ giúp em xác định chính xác số n để so sánh. Tức ví dụ nếu em có 2*pi thì mô tả dưới lệnh thành phép toán em có `(2*pi/pi)` sẽ là 2 và đem chúng so sánh với `round((2*pi/pi),4)` thì sau lệnh `round((2*pi/pi),4)` sẽ là 2 và thỏa; còn trường hợp em có temp_1 là 10 thì có `(10/pi) = 3.183098` ở dạng chưa làm tròn thì khi qua lệnh làm tròn 4 chữ số nó sẽ là 3.1831 và đã khác nhau nên trường hợp này loại. Sở dĩ em dùng cách này vì ta biết pi có độ dài rất lớn, mà số em đang xét đây là làm tròn 4 chữ số nên trong mọi trường hợp khi dữ liệu không phải dạng n*pi mà truyền vào thì hàm này sẽ không kiểm tra vì nếu số n thông thường khi chia cho pi sẽ ra một dãy số lẻ lớn, và ta làm tròn 4 chữ số thập phân thì chúng sẽ khác nhau, khi là n*pi thì khi chia cho pi sẽ mất pi còn n, và đó là nguyên nhân duy nhất khiến cho hai hàm `(temp_1/pi)` và `round((temp_1/pi),4)` có thể bằng. Tiếp theo thì em

cần cho điều kiện của mình không được có không vì nếu có không thì kết quả không còn chính xác nữa theo ta biết $0/\pi$ vẫn là 0 nên em cần dùng lệnh `&&` - tức and để loại đi trường hợp bằng 0. Sau khi xác định đúng dạng t thì có $t = \text{numel}(\text{num2str}(\text{temp_1}/\pi))$ thì ở đây em đã xác định đúng dạng $n*\pi$ rồi, nên bây giờ em sẽ tách ra n để xác định xem n có bao nhiêu chữ số, mà nếu dùng `numel` thông thường thì sẽ chỉ đếm 1 vì `numel` là câu lệnh đếm số phần tử trong mảng nên ở đây em đã thay đổi kiểu dữ liệu thành kiểu string để với số n bất kì có được em sẽ ra được số ký tự tương ứng và số ký tự này sẽ là số chữ số của n , sau khi có được số chữ số n thì ta biết biểu thức người dùng nhập vào sẽ có dạng “ $n*\pi$ ”, mà n đã xác định số chữ số rồi, chỉ còn “ π ” mà ta thấy “ π ” sẽ là 3 ký tự nên ta có $t = t + 3$. Sau khi kết thúc lệnh này thì với n âm hay dương bất kì miễn là số thập phân làm tròn sau dấu phẩy bốn chữ số thập phân mà có dạng $n*\pi$ thì ta đều có thể có được vị trí của dấu chấm phẩy. Tiếp theo thì nếu không rơi vào trường hợp π mà là trường hợp thông thường thì ta sẽ dùng lệnh $t = \text{numel}(\text{num2str}(\text{temp_1}))$, vì câu lệnh này sẽ cho ra số lượng của mảng, nên khi chuyển từ kiểu dữ liệu số sang kiểu ký tự nên sẽ có mỗi vị trí trong mảng `temp_1` sẽ là 1 ký tự. Ví dụ em có số -12 thì khi kết thúc câu lệnh $t = \text{numel}(\text{num2str}(\text{temp_1}))$ ta sẽ có được biến $t = 3$ sau câu lệnh trên. Như vậy là em đã thu được số lượng của số ở vị trí đầu tiên trong mảng, sau khi có được thì em sẽ cộng cho 1 là bỏ qua vị trí của dấu mở ngoặc vuông, sau đó cộng tiếp cho 1 để đến vị trí để kiểm tra dấu chấm phẩy nên em có lệnh: $t = t + 2$. Sau đó em sẽ cho mảng `data` chạy đến vị trí t , tại đó em sẽ kiểm tra xem ký tự tại `data(t)` có phải là dấu chấm phẩy hay không qua câu lệnh: `if ((data(t) ~= ";"))`. Câu lệnh này sẽ kiểm tra xem nếu ký tự tại `data(t)` không phải là dấu chấm phẩy thì sẽ thực hiện lệnh ở trong, em sẽ đảm bảo là `data(t)` sẽ không phải là số dữ liệu mà người dùng nhập vào thông qua các câu lệnh ở trên nên sẽ đảm bảo chính xác khi bóc ký tự ra và kiểm tra. Những câu lệnh tiếp theo trong lệnh `if` thì sẽ tương tự hoàn toàn với phần bắt lỗi khi người dùng nhập sai cú pháp là có dấu ngoặc vuông; nên ở đây em xin phép không giải thích lại ạ. Và đó là lệnh kiểm tra dấu chấm phẩy, nếu không thỏa – tức có dấu chấm phẩy rồi, thì sẽ rẽ nhánh xuống lệnh `else`, ở lệnh `else` em sẽ cho `check_temp1 = 1`. Như vậy sau khi kiểm tra xong hai điều kiện là có dấu ngoặc vuông và có dấu chấm phẩy thì em còn điều kiện là $N > 0$ và $a \neq b$. Việc kiểm tra từng phần giúp đảm bảo chương trình của em bắt đúng lỗi và chặt chẽ hơn. Tiếp theo là phần code của điều kiện cuối cùng trong hàm tự bắt lỗi của em. Sau đây là hình mô phỏng khi nhập vào khoảng tính tích phân khác cú pháp `[a;b]:`



➤ Code bắt lỗi khi người dùng nhập a bằng b và $N \leq 0$

```
data_1 = str2num(data);
a = data_1(1);
b = data_1(2);
N = str2num(app.SNonconEditField.Value);
if ((N<=0) || Length(N) >1)
    msg = {'Lỗi xác định số N đoạn con chia nhỏ', 'N chỉ là 1 số duy nhất', 'N không thể bằng 0 và không thể âm'};
    msgbox(msg, 'Error', 'error');
else check_temp2 = 1;
end
if (a==b)
    msg = {'Lỗi xác định khoảng tính tích phân', 'a và b không thể bằng nhau'};
    msgbox(msg, 'Error', 'error');
else check_temp3 = 1;
endcheck_1 = check_temp & check_temp1 & check_temp2 & check_temp3;
```


➤ *Giải thích code*

Ở phần này đầu tiên thì em cần biến đổi từ kiểu dữ liệu string sang số của biến data – data được lưu từ ô app.KhongtnhtchphnEditField. Tiếp theo lệnh str2num dùng để thực hiện việc chuyển đổi từ kiểu dữ liệu chuỗi sang số. Ví dụ nếu em nhập đúng cú pháp có dạng là [1;2] thì khi qua lệnh str2num sẽ thành một ma trận có 2 dòng, 1 cột gồm dòng 1 là số 1 và dòng 2 là số 2. Nắm được điều đó thì em sẽ có biến tạm tên data_1 dùng để tạo 1 mảng nhằm lưu lại các số sau chuyển đổi. Tiếp theo thì em có lệnh a = data_1(1) tức là em đang lấy dữ liệu đang lưu ở data_1 ở vị trí thứ nhất gán vào a, đây sẽ là a trong [a;b]. Tiếp theo thì em có b = data_1(2) tức là em sẽ lấy số nằm ở vị trí thứ hai trong mảng data_1 ra và lưu vào b, b sẽ là b trong khoảng tính tích phân [a;b]. Lý do em không ghi b = data_1(2,1), a = data_1(1,1) vì đây sẽ chỉ là mảng một chiều và chiều của chúng theo cột, em ghi vô cho đầy đủ cũng sẽ có kết quả tương tự như em chỉ ghi là data_1(1) và data_1(2) nên ở đây em ghi rút gọn. Sau khi kiểm tra đúng cú pháp thì MATLAB tiến hành thu nhận dữ liệu gồm khoảng tích phân – đã giải thích ở trên và nhận dữ liệu từ ô app.SNonconEditField, và biến N dùng để lưu, cấu trúc code là: N = str2num(app.SNonconEditField.Value) được người dùng nhập từ UI. Và tương tự như này giờ em có trình bày thì để đọc giá trị ra và lưu vào biến thì ta cần dùng lệnh .Value ở sau ô đó để tiến hành lấy giá trị và lưu vào N. Ở đây ta có ô nhập vào có kiểu là Text – tức là dữ liệu đang được xem là string bất chấp ta có nhập số thì MATLAB vẫn đang hiểu là từng ký tự nên ta cần có lệnh str2num để đổi từ ký tự sang số nhằm đọc ra giá trị, đó là lý do em có câu lệnh str2num. Và ở giai đoạn này thì đồng thời ta cũng sẽ kiểm tra xem số đoạn con N có được nhập hay không, nếu không được nhập thì MATLAB sẽ bắt lỗi. Nếu đọc N từ giao diện người dùng thì ta sẽ có được N kiểm tra xem có âm hay không, và cũng như người dùng có nhập nhầm N là khoảng tính tích phân thì ta sẽ kiểm tra và báo lỗi. Nếu N âm hoặc N bị nhầm khoảng tính tích phân hoặc thỏa cả 2 thì ta sẽ thực hiện lệnh ở trong thì ta sẽ thực thi xuất ra màn hình các dòng cảnh báo như sau: . Chương trình gồm tạo một cell array gồm 3 câu sẽ được lưu và lần lượt lấy ra ở msgbox, 3 câu đó là: “ Lỗi xác định số N đoạn con chia nhỏ ”; “ Số đoạn con N chia nhỏ không thể bằng 0 và không thể âm ” và câu: “ Vui lòng nhập lại số đoạn con N >0 ”, các câu này sẽ lần lượt hiện ra theo từng hàng, hàng thứ nhất là câu đầu tiên em ghi, câu thứ hai là câu tiếp theo và câu thứ ba là câu cuối cùng. Tiêu đề của hộp thoại tin nhắn là Error, và icon lỗi error tương tự như

giải thích ở phần trên. Hình bên là chạy thử nếu nhập $N \leq 0(-1)$ và N bị nhầm là khoảng tính tích phân

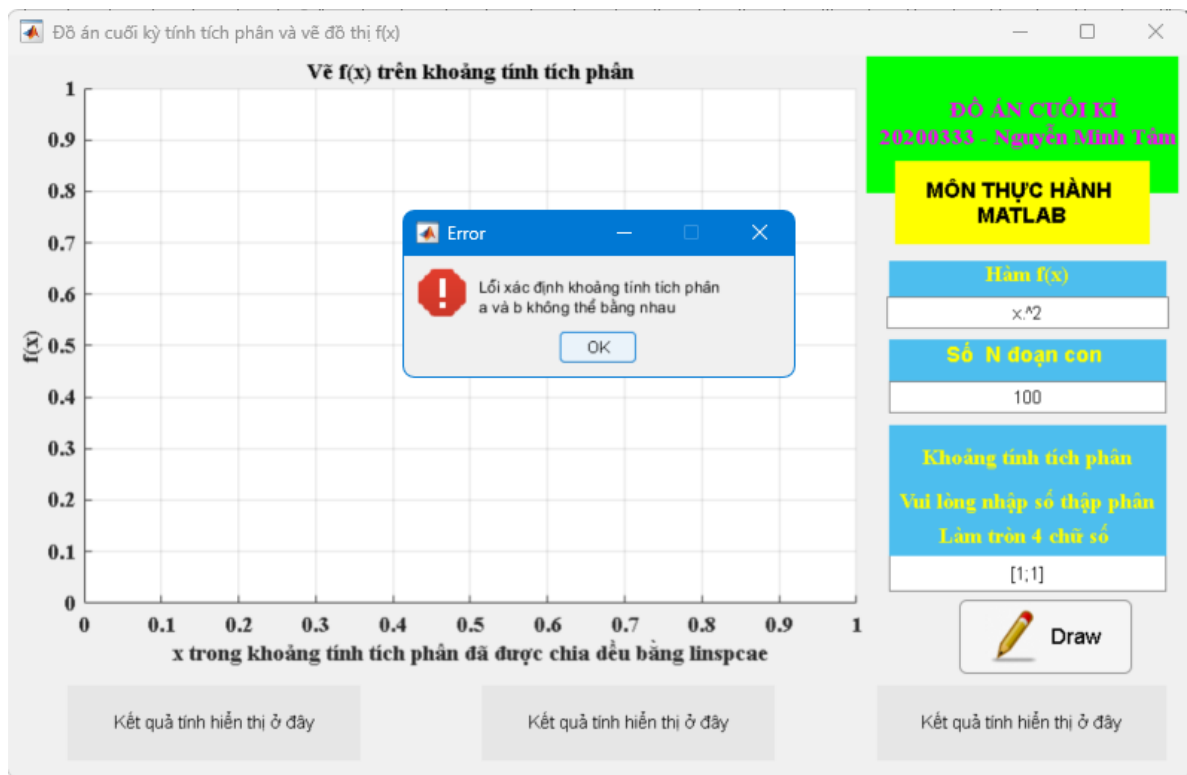


Trường hợp nhập số N đoạn con nhiều hơn 1



Trường hợp số đoạn con là -1

Tiếp theo thì sau khi đọc ra số đoạn con N và kiểm tra thì em sẽ tiến hành kiểm tra a có bằng b hay không thông qua lệnh: `if (a==b)`. Nếu a và b bằng nhau thì sẽ thực hiện việc tạo một cell array gồm hai câu tin nhắn thông báo lỗi, câu thứ nhất là: Lỗi xác định khoảng tính tích phân – câu dẫn tên lỗi, câu thứ hai là: a và b không thể bằng nhau – là câu mô tả chi tiết lỗi. Nếu a và b ta nhập mà bằng nhau thì sẽ có hộp thoại như hình dưới:



3.2.b/ Code giao diện hiển thị kết quả

➤ Code

```
if (check_1 == 1)
    format long;
    results = tichphanhinhthang(fx,a,b,N);
    results_Simpson = tichphanSimpson(fx,a,b,N);
    results_chinhxac = tichphanChinhxac(fx,a,b);
    text_2 = sprintf('Phương pháp tích phân hình
thang\n%.10f',results);
    app.KtqutnhhinthyLabel_2.Text = text_2 ;
    text_1 =  sprintf('Phuong pháp tích phân Simpson
\n%.10f',results_Simpson);
    app.KtqutnhhinthyLabel.Text = text_1 ;
    text_3 =  sprintf('Phuong phap tích phân Chính
xác\n%.10f',results_chinhxac);
    app.KtqutnhhinthyLabel_3.Text = text_3 ;
    x_divide = linspace(a,b,10);
    plot(app.UIAxes,x_divide,fx(x_divide),'LineWidth',3);
else
    text =  sprintf('Kết quả tính hiển thị ở đây');
    app.KtqutnhhinthyLabel_3.Text = text ;
    app.KtqutnhhinthyLabel_2.Text = text ;
    app.KtqutnhhinthyLabel.Text = text ;
    x_divide = linspace(0,1,10);
    plot(app.UIAxes,x_divide,0);
end
```

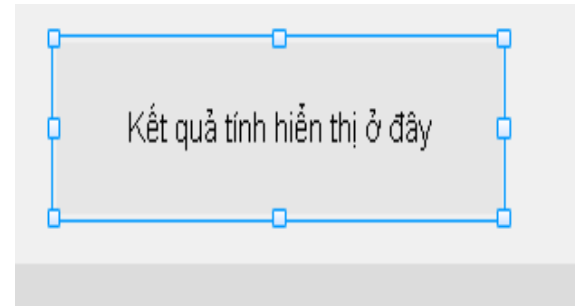
➤ Giải thích code

Sau khi bắt lỗi một số trường hợp mà MATLAB không thể tự bắt lỗi, thì ở đây em trình bày code để xuất kết quả ra màn hình ứng dụng. Khi không còn lỗi thì biến check_1 của em sẽ có giá trị là 1, và em sẽ cho đoạn code hiển thị ra kết

quả tính toán nếu biến check_1 của em là 1 nên em có dòng lệnh if(check_1 == 1). Tiếp theo thì để có thể xuất ra số thập phân sau dấu phẩy có 10 chữ số thì em cần phải format long, tức là định dạng kiểu dữ liệu của em là long để cho ra lên đến 15 chữ số thập phân - ở đây đề yêu cầu là 10 nên em sẽ trình bày ở giải thích code sau. Tiếp theo thì em sẽ đặt các biến nhằm gọi các function ra để tính toán, lúc đầu thì bên cạnh việc bắt lỗi thì em cũng đã đọc những thông số mà người dùng đã nhập để thực hiện tính toán nên sau khi qua được giai đoạn bắt lỗi thì ta đang có là: khoảng tích phân với giá trị được lưu từ biến a và b, số đoạn con N và hàm f(x). Các biến lưu các phương pháp là: results là kết quả tính gần đúng tích phân hình thang với input là fx, a, b và N nên ta gọi function là results = tichphanhinhthang(fx,a,b,N); tiếp tjeo là results_Simpson là tính gần đúng tích phân Simpson với input sẽ là fx, a, b và N nên ta có lệnh: results_Simpson = tichphanSimpson(fx,a,b,N); cuối cùng là lưu kết quả tính chính xác với biến lưu là results_chinhxac, input là fx, a và b vì ở phương pháp tính chính xác mình chỉ cần theo đúng công thức: $\int_a^b f(x)$ nên em có dòng gọi function là: results_chinhxac = tichphanChinhxac(fx,a,b). Tiếp theo là để in kết quả ra label thì có rất nhiều cách để in ra label, em chọn cách là soạn dòng thông báo ra lưu vào biến tạm, sau đó dùng hàm “.Text” của app designer matlab để in ra màn hình. Giải thích cụ thể em có đầu tiên có lệnh: text_2 = sprintf('Phương pháp tích phân hình thang\n%.10f',results), lệnh sprintf dùng để định dạng các dạng gồm chuỗi, số thành một kiểu dữ liệu duy nhất là string – chuỗi, nó được lưu vào biến text_2, tiếp theo là dòng hiện ra kết quả là: “Phương pháp tích phân hình thang” và để xuống dòng ghi ra kết quả thì em có lệnh “\n” in ra màn hình kết quả thông qua lệnh “%.10f” và gọi đến biến results – đang lưu là tính gần đúng tích phân hình thang, giải thích lệnh thì “%.10f” dùng để mình định dạng là sau dấu phẩy lấy 10 chữ số thập phân, “%f” là kiểu float. Sau khi em có biến text_2 để lưu các dòng in ra label thì em sẽ tiến hành xuất ra label thông qua lệnh: app.KtqutnhhinthyLabel_2.Text = text_2; giải thích code thì em có tên của label để in ra àn hình sẽ là app.KtqutnhhinthyLabel_2, hàm Text là hàm thay đổi text của label trong matlab, em dùng cái đó để có thể thay đổi dòng text, dòng text mới sẽ theo biến text_2 – là biến em dùng để lưu tạm các dòng text in ra label mà ở trên em đã trình bày. Kết quả in ra màn hình là:

Phương pháp tích phân hình thang
0.1796515769

Để xuống hàng thì em dùng lệnh “\n” và để có thể có là cân xứng hai bên, ở giữa thì em sẽ chỉnh HorizontalAlignment, VerticalAlignment, sau đó em sẽ chỉnh khung cho text như hình dưới đây để có thể in ra kết quả như hình em đã chụp



Tương tự như thế thì em có biến text_1 dùng để lưu dòng in ra kết quả màn hình kết quả tính gần đúng tích phân Simpson, thông qua lệnh `text_1 = sprintf('Phương pháp tích phân Simpson \n%.10f',results_Simpson)`. Em có hàm `sprint` nhằm để lưu chuỗi gồm câu thông báo là: “Phương pháp tích phân Simpson”, xuống dòng là “\n” và ghi ra kết quả là: “%.10f” thông qua lệnh xuất `results_Simpson`. Sau khi em đã có dòng để in ra kết quả đang được lưu dưới tên là text_1 thì em sẽ tiến hành ghi ra label, tên label của em đang là: `app.KtqutnhhinthyLabel`, để ghi ra label thì có lệnh: `app.KtqutnhhinthyLabel.Text = text_1`, hàm `Text` là hàm thay đổi dòng text của label sau lệnh này thì ta sẽ ghi kết quả ra label như hình dưới đây:

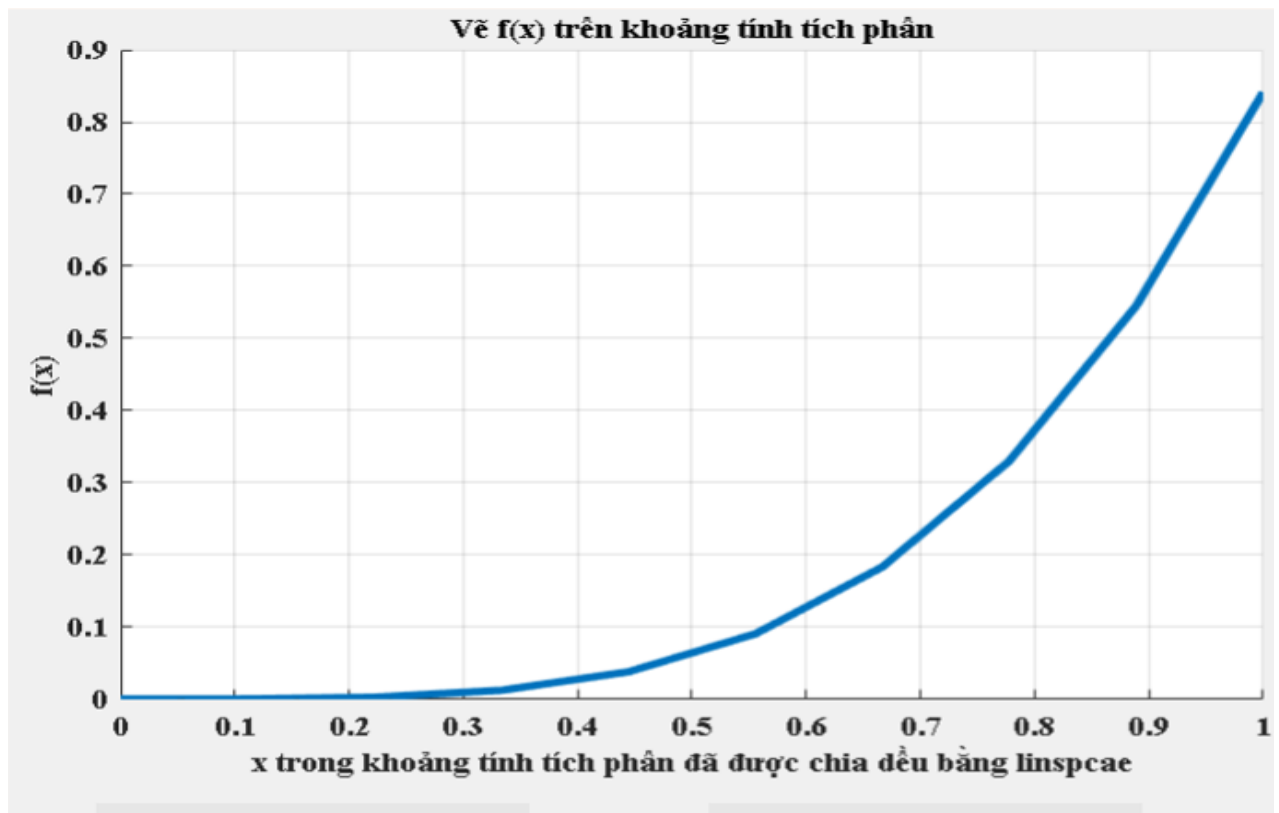


Để được định dạng trên em đã giải thích ở trên. Cuối cùng là ta cần ghi ra label kết quả chính xác, dòng text để hiển thị kết quả qua code: `text_3 = sprintf('Phương pháp tích phân Chính xác\n%.10f',results_chinhxac)`, tức biến text_3 đang là biến lưu chuỗi hiển thị kết quả. Em có hàm `sprint` nhằm để lưu chuỗi gồm câu thông báo là: “Phương pháp tích phân chính xác”, xuống dòng là “\n” và ghi ra kết quả là: “%.10f” thông qua lệnh xuất `results_chinhxac`. Sau khi em đã có dòng để in ra kết quả đang được lưu dưới tên là text_1 thì em sẽ tiến hành ghi ra label, tên label của em đang là: `app.KtqutnhhinthyLabel_3`, để ghi ra label thì có lệnh: `app.KtqutnhhinthyLabel_3.Text = text_3`, sau lệnh này thì ta sẽ ghi kết quả ra label như hình dưới đây:

Phương pháp tích phân Chính xác

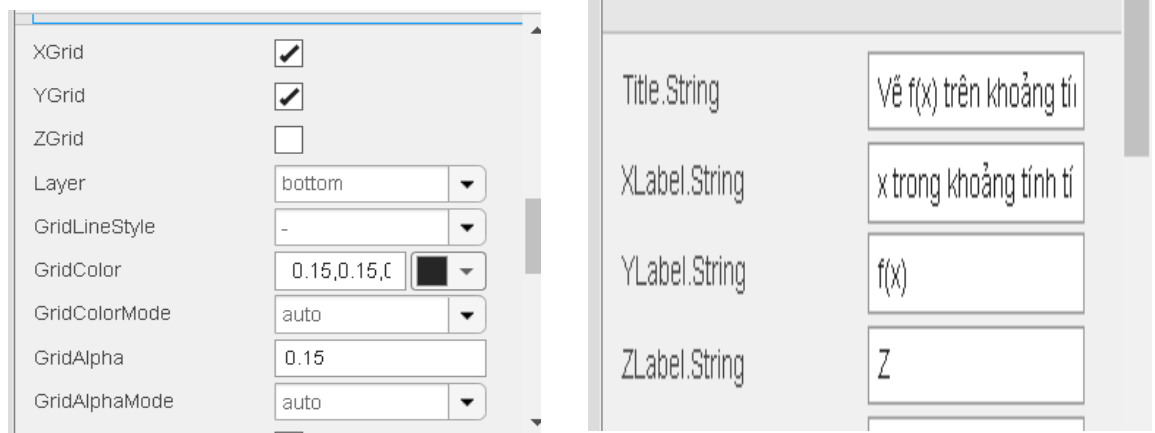
0.1770985749

Sau khi in kết quả ra label thì em sẽ tiến hành dùng câu lệnh `linspace` để chia đều khoảng tính tích phân từ $a \rightarrow b$ thành 10 đoạn nhỏ. Em có lệnh: `x_divide = linspace(a,b,10)`. Sau khi kết thúc câu lệnh này em sẽ có `x_divide` sẽ chứa 10 phần tử là vector được chia đều từ a đến b . Ví dụ em có $a = 0$; $b = 1$ thì sau câu lệnh `x_divide` sẽ là: `[0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]`. Sau khi chia đều khoảng tính tích phân thì em sẽ vẽ $f(x)$ bằng lệnh: `plot(app.UIAxes,x_divide,fx(x_divide),'LineWidth',3)`, đồ thị có tên là `app.UIAxes`, vẽ theo $f(x)$ mà x ở đây sẽ là x sau khi chia nên sẽ là `x_divide`, sau khi chia đều thì em tiến hành thế vào hàm $f(x)$ nên em có lệnh `f(x_divide)`. Để vẽ cho dễ nhìn thì em sẽ có độ dày đường vẽ là 3, thông qua lệnh (`'linewidth',3`). Sau khi kết thúc lệnh thì em sẽ có kết quả là:

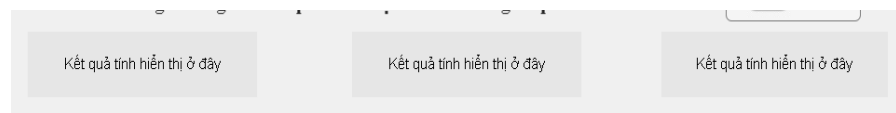


Sau khi vẽ xong thì em đã hoàn thành các yêu cầu của đề bao gồm in ra kết quả của cả ba phương pháp tính ra label, in kết quả sau dấu phẩy là 10 chữ số thập

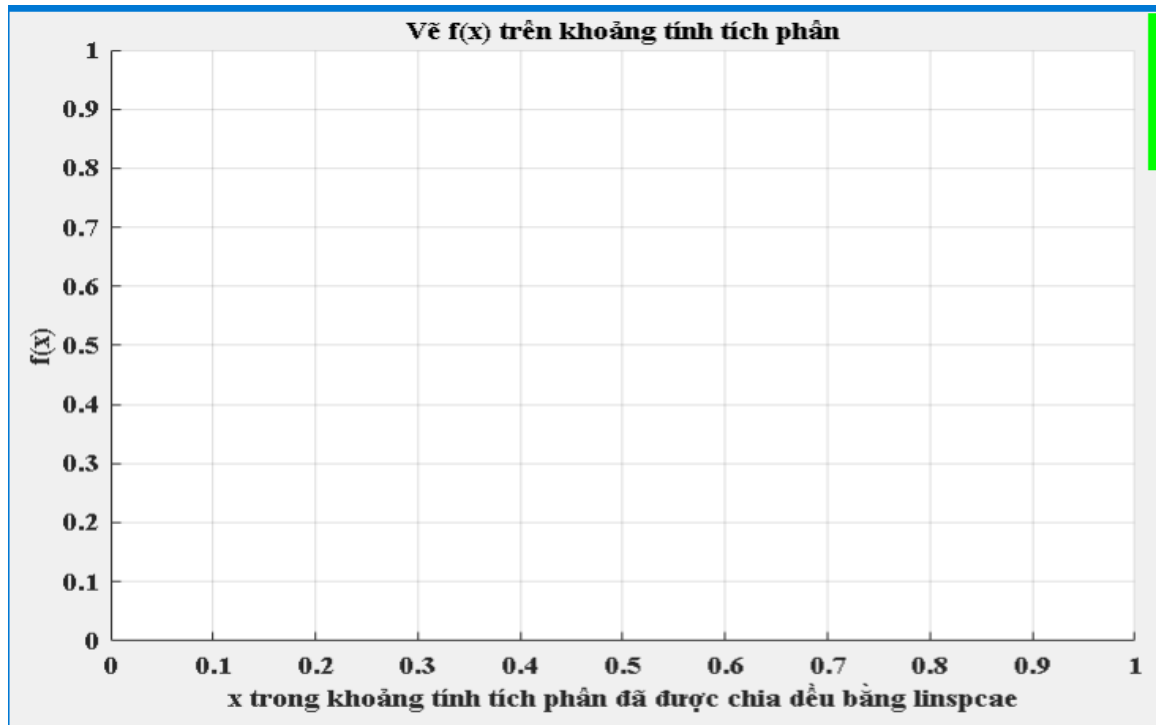
phân và vẽ $f(x)$ trong khoảng tính tích phân. Để có được đồ thị gồm các ô lưới và chú thích như trên thì em đã chỉnh ở phần design như sau:



Và đó là những dòng hiển thị kết quả, sau khi hiển thị kết quả khi không sai thì ở đây em cần reset nếu thỏa điều kiện sai của hàm em code tự bắt lỗi. Sau đây là những giải thích code khi cần reset kết quả nếu $\text{check_1} \neq 0$. Đầu tiên thì em sẽ đưa label của mình về mặc định khi mới khởi động bằng câu lệnh: `text = sprintf('Kết quả tính hiển thị ở đây')`, mặc định ban đầu như em đã thêm vào tấm ảnh tổng quan em đã thiết kế thì khi bấm Run thì label của em mặc định hiển thị: “Kết quả tính hiển thị ở đây”, nên em đã cho chúng hiện lại dòng text như vậy mỗi khi có lỗi, lỗi nhập không đúng định dạng hay nhập bị sai. Có 3 label của app nên em sẽ gọi 3 ô đó ra gồm `app.KtqtnhnhinthyLabel_3`, `app.KtqtnhnhinthyLabel_2`, `app.KtqtnhnhinthyLabel` và truyền cùng 1 text đang được lưu ở biến `temp`. Như vậy dùng cách này giúp em có thể chạy code được nhiều lần, không có lỗi thì chạy tốt còn có lỗi thì app sẽ báo lỗi và reset về mặc định như hình sau:



Sau khi reset của label thì em tiến hành reset về mặc định của đồ thị $f(x)$, bằng cách chia lại `linspace` của `x_divide` là từ 0 đến 1 thành 10 và vẽ theo $y = 0$, giới hạn trục tung là từ 0 đến 1 thông qua `ylim(app.UIAxes,0:1)`, em sẽ chọn mục tiêu để giới hạn trục tung ở đồ thị có tên là `app.UIAxes`, `0:1` là chạy từ 0 đến 1 để có thể giống với như lúc ta thiết kế ban đầu. Sau khi chạy và biến `check_1` bằng 0 thì ta ra như mặc định đúng theo vừa mới kéo vô như hình sau:



Sau khi thực hiện hàm giao diện cho cả trường hợp cả khi đúng và sai thì chương trình của em sẽ chạy ổn định, các lỗi sẽ được xuất ra màn hình. Tuy vậy nhược điểm của code của em sẽ có khuyết điểm là không tính được bình phương, căn bậc ba, căn bậc hai và phân số khi mà người dùng vẫn chủ động muốn nhập khoảng tính là số bình phương, khi tính giá trị không phải là số thập phân (ngoại trừ pi) thì buộc phải nhập giá trị sau khi lấy số thập phân. Đó là phần em code đối với các lỗi mà em đã lập trình cho MATLAB không thể tự động bắt lỗi, hầu hết các lỗi thì do không sai cú pháp, không phải là lệnh MATLAB không hiểu mà là những lỗi ảnh hưởng kết quả chính xác, ảnh hưởng đến yêu cầu về bài toán nên em đã lập trình ở giữa trong cấu trúc try catch. Còn phần MATLAB tự bắt lỗi được thì hầu hết đều là lỗi cú pháp, lỗi MATLAB không thể hiểu để thực thi lệnh. Sau đây là phần code lỗi cho MATLAB có thể bắt được nó nằm trong catch error.

3.2.c/ Phần code bắt các lỗi mà MATLAB có thể bắt được

➤ Code:

```
catch error

    switch(error.identifier)

        case'MATLAB:m_incomplete_statement'

            msgbox('Vui lòng nhập hàm f(x)', 'Error', 'error');

        case'MATLAB:mpower:notScalarAndSquareMatrix'

            msg = {'Lỗi nhân ma trận!', 'Đề nhân từng phần tử, dùng dấu "."  
trước lệnh.  '};

            msgbox(msg, 'Error', 'error');

        case'MATLAB:m_missing_operator'

            msgbox('Sai định dạng hàm , vui lòng nhập lại', 'Error', 'error');

        case'MATLAB:UndefinedFunction'

            msg = {'Lỗi không xác định hàm!', 'Chỉ nhập hàm một biến và hàm  
f(x) không được chứa ký tự đặc biệt  '};

            msgbox(msg, 'Error', 'error');

        case'MATLAB:badsubscript'

            if (isempty(data) && isempty(N))

                msg = msg = {'Vui lòng nhập khoảng tính tích phân', 'Vui lòng  
nhập vào số đoạn con N'};

            elseif isempty(data)

                msg = {'Vui lòng nhập khoảng tính tích phân'};

                msgbox(msg, 'Error', 'error');

            elseif isempty(N)

                msg = {'Vui lòng nhập vào số đoạn con N'};

                msgbox(msg, 'Error', 'error');

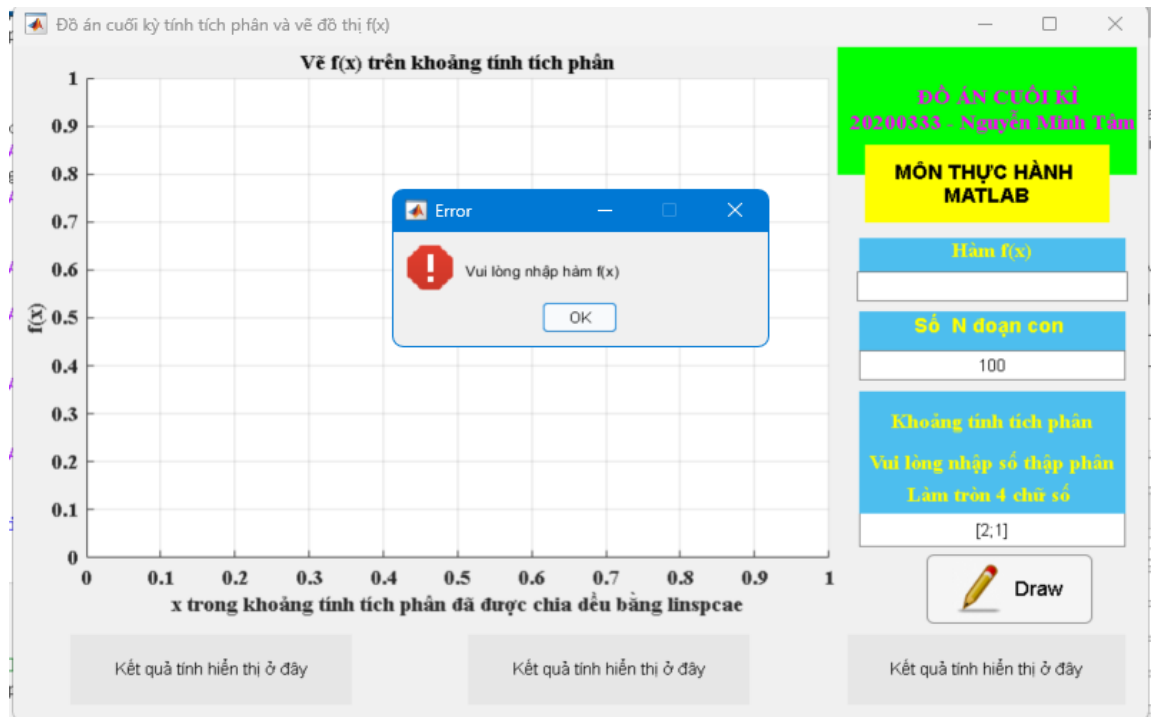
            end

        otherwise msgbox({error.identifier,error.message}, 'Error', 'error');

    end    end
```

➤ *Giải thích code*

Đầu tiên thì hàm MATLAB có catch error, tức là trường hợp mà MATLAB sẽ thực thi lệnh ở sau catch error nếu có lỗi, còn các chương trình thực thi lệnh khi không có lỗi thì sẽ thực thi code nằm ở giữa try và catch. Đầu tiên thì em sẽ dùng cấu trúc switch case để có thể rẽ nhánh chính xác đến nơi cần được xuất ra dòng cảnh báo. Ở đây ta có thể dùng cấu trúc if else nhưng ở đây vì không cần phải xét điều kiện theo logic, sở dĩ ở đây chỉ cần rẽ nhánh theo từng trường hợp khi error.identifier có sự mô tả lỗi theo từng tên lỗi, case theo từng tên lỗi đó ta sẽ ra được những dòng lệnh để có thể đưa ra hộp thoại pop – up cảnh báo. Tiếp theo là em sẽ nói trường hợp đầu tiên khi error.identifier đưa ra lỗi là “MATLAB:m_incomplete statement” tức ở đây là lỗi không thể hoàn thành câu lệnh – tức định dạng hàm mà MATLAB không hiểu, có thể do chưa nhập hàm $f(x)$. Dòng thông báo lỗi sẽ có một dòng là: “Vui long nhập hàm $f(x)$ ”. Còn các thông số gồm: “Error” là dòng tiêu đề hộp thoại cảnh báo, “error” là icon cảnh báo có lỗi mà em đã giải thích chi tiết ở trên. Sau đây là dòng cảnh báo khi MATLAB bắt được trường hợp lỗi đầu tiên:



Trường hợp tiếp theo là khi error.identifier bắt được lỗi là “MATLAB:mpower:notScalarAndSquareMatrix” tức là lỗi nhân ma trận tức là khi nhân vô hướng – không phải là lấy hai ma trận nhân nhau thì ta phải dùng dấu “.” trước lệnh nhân, mũ, chia để có thể thực hiện phép tính vô hướng, sẽ không còn lỗi. Dòng thông báo lỗi sẽ có hai dòng, em sẽ tạo một cell array gồm dòng 1 là: “Lỗi nhân ma trận” và dòng 2 là: “Để nhân từng phần tử, dùng dấu “.” trước lệnh” thì em sẽ truyền lần lượt dòng 1 và dòng 2 vào lệnh msgbox để in ra hộp thoại, do truyền

2 dòng nên hộp thoại sẽ in ra dòng 1 trước, dòng 2 sau. Còn các thông số gồm: “Error” là dòng tiêu đề hộp thoại cảnh báo, “error” là icon cảnh báo có lỗi mà em đã giải thích chi tiết ở phần code cho trường hợp bắt lỗi mà MATLAB không thể tự động bắt. Sau đây là kết quả mô phỏng khi nhập vào phép tính không phải là nhân vô hướng - không có dấu chấm ở trước:



Tiếp theo thì em sẽ xét trường hợp mà error.identifier bắt lỗi là “m_missing_operator” là lỗi khi hàm có ký tự đặc biệt và MATLAB không thể hiểu được ký tự đó là gì. Dòng thông báo lỗi sẽ có một dòng là: “Sai định dạng hàm” lệnh msgbox để in ra hộp thoại, in ra hộp thoại. Còn các thông số gồm: “Error” là dòng tiêu đề hộp thoại cảnh báo, “error” là icon cảnh báo có lỗi mà em đã giải thích chi tiết ở phần code cho trường hợp bắt lỗi mà MATLAB không thể tự động bắt. Sau đây là phần mô phỏng với hàm nhập có ký tự đặc biệt:



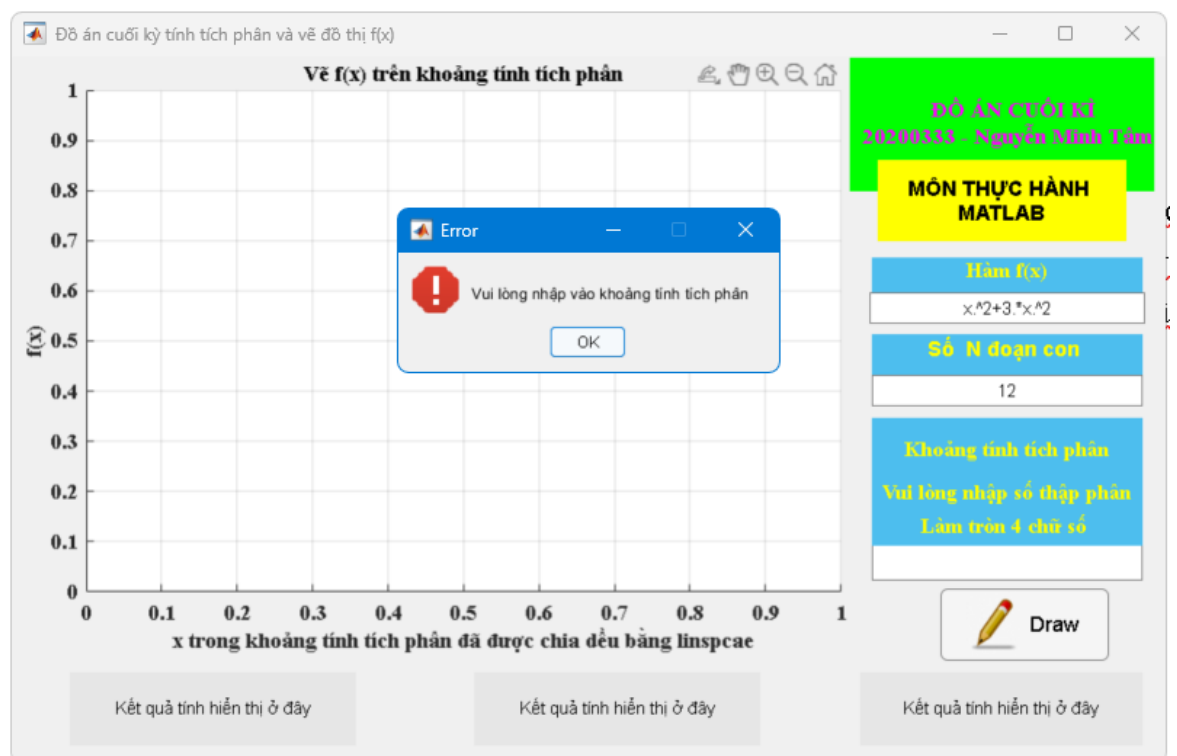
Tiếp theo là trường hợp mà `error.identifier` trả về lỗi là: “MATLAB:UndefinedFunction” tức là lỗi mà MATLAB không thể xác định hàm nhập tức là ở đây có thể có hai ẩn, hoặc ẩn mà ta nhập nó không phải là x mà là ký tự khác nên MATLAB không thể hiểu. Dòng thông báo lỗi sẽ có hai dòng, em sẽ tạo một cell array gồm dòng 1 là: “Lỗi không xác định hàm!” và dòng 2 là: “Chỉ nhập hàm một biến và hàm cần theo biến x tức $f(x)$ ” thì em sẽ truyền lần lượt dòng 1 và dòng 2 vào lệnh `msgbox` để in ra hộp thoại, do truyền 2 dòng nên hộp thoại sẽ in ra dòng 1 trước, dòng 2 sau. Còn các thông số gồm: “Error” là dòng tiêu đề hộp thoại cảnh báo, “error” là icon cảnh báo có lỗi mà em đã giải thích chi tiết ở phần code cho trường hợp bắt lỗi mà MATLAB không thể tự động bắt. Sau đây là kết quả mô phỏng khi nhập vào phép tính bị thừa biến:



Đó là tất cả những lỗi mà hàm $f(x)$ có thể có, sau đây sẽ là những lỗi mà khi người dùng quên nhập số đoạn con N hay khoảng tính tích phân hay là quên nhập cả hai. Lỗi này thì được `error.identifier` có lỗi là: “MATLAB:badsubscript” tức là lỗi không thể nào bằng 0 khi đọc dữ liệu. Em có 3 trường hợp, và để kiểm tra xem ô nào trống thì em dùng hàm `isempty` – là lệnh sẽ trả về 1 khi hàm này không có dữ liệu được nhập, sau đây là liệt kê của em về ba trường hợp. Thứ nhất là cả hai ô đều trống bằng lệnh: `if (isempty(data) && isempty(N))`, khi thực thi dòng cảnh báo thì em sẽ có các dòng lệnh là: dòng 1 là: “Vui long nhập khoảng tính tích phân”, dòng 2 là: “Vui lòng nhập vào số đoạn con N ” và sẽ được truyền lần lượt vào lệnh xuất ra hộp thoại bằng lệnh `msgbox` để có thể đưa ra màn hình theo thứ tự là dòng 1 và dòng 2. Sau đây là mô phỏng nếu không nhập cả số đoạn con và khoảng tính tích phân:



Thứ hai là lỗi nếu đã nhập số đoạn con nhưng chưa nhập khoảng tính tích phân. Đối với trường hợp này em dùng lệnh: `elseif isempty(data)` để có thể rẽ nhánh chính xác, và em sẽ bớt đi một dòng sau đó in ra màn hình. Sau đây là mô phỏng:



Thứ ba là ngược lại với trường hợp thứ hai – tức là khoảng tính tích phân đã có nhưng không có số N đoạn con chia nhỏ. Cũng như trường hợp 2 ta sẽ có lệnh: `elseif isempty(N)`, và so với trường hợp 1 thì ta chỉ cần loại bỏ dòng thông báo lỗi trường

hợp thứ hai là ta sẽ thu được dòng thông báo lỗi ở trường hợp ba này. Sau đây là mô phỏng của em:



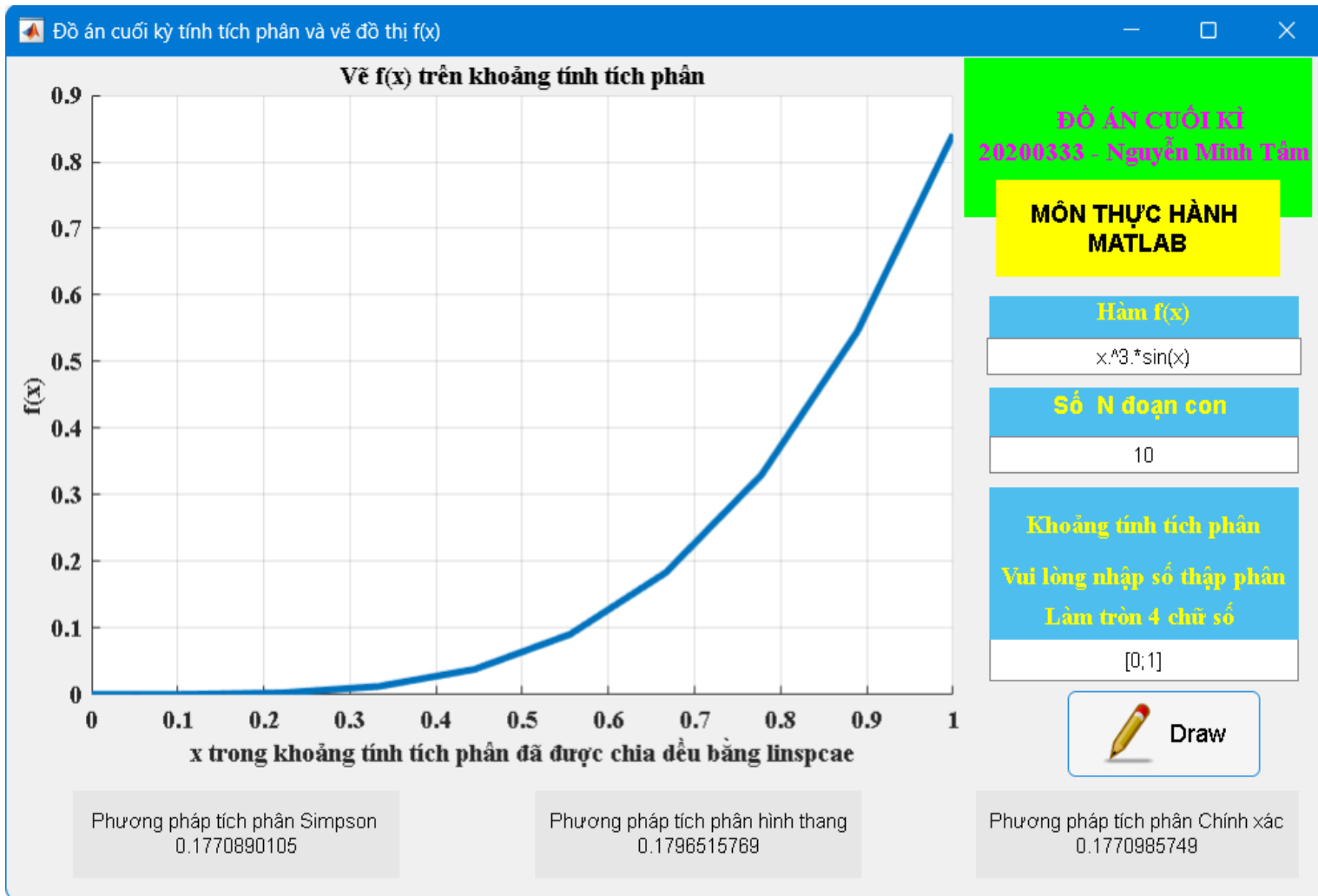
Sau các lỗi mà trong quá trình làm có phát sinh mà em đã chỉnh sửa thì các lỗi thông thường đều được phát hiện và thông báo cho người dùng. Tuy vậy trong quá trình sử dụng sẽ có những lúc có lỗi mà chưa được em gọi tên đến nên em sẽ để lệnh `otherwise` – tức là các trường hợp không được liệt kê thì sẽ được gọi tên lỗi và mô tả chi tiết thông qua lệnh: `msgbox({error.identifier,error.message},'Error','error')`, tức ở đây sẽ có: “error.identifier” là các lỗi mà MATLAB trả về, gọi tên lỗi của MATLAB, tiếp theo là “error.message” là dòng mô tả lỗi và cũng như gợi ý của MATLAB để ta có thể hiểu và sửa cho đúng. Còn “Error” là tiêu đề của dòng báo lỗi còn “error” là dòng để có icon báo lỗi mà em đã trình bày chi tiết ở trên. Sau khi kết thúc phần các lỗi khi nhập khoảng tính tích phân và số đoạn con N mà đã trình bày ở trên thì hầu như các lỗi mà MATLAB tự bắt hay lỗi do cú pháp, lỗi ảnh hưởng kết quả đều được em trình bày Ở phần code em đã để hai lệnh `end` cùng một hàng và lệnh `otherwise` cùng 1 hàng vì code của em khá dài nên một trang không đủ nên em viết cùng 1 hàng để có thể viết cùng 1 hàng không bị ngắt. Sau đây là phần tổng kết của em về đồ án.

4. Tổng kết đồ án

- Em đã hoàn tất đồ án và phần thiết kế giao diện dễ nhìn và sinh động
- Các lỗi trong quá trình kiểm thử mà xuất hiện thì em đã giải quyết hết cả, đảm bảo khi nhập sai ở dữ liệu nào thì sẽ được thông báo lỗi và nhắc nhở nhập sao cho đúng
- Nhược điểm đồ án: Do kiến thức hạn chế nên em vẫn chưa có thể cấu hình sao cho với phân số, số căn bậc hay số mũ nào thì ứng dụng của em vẫn chạy, điều này là thiếu

xót của đồ án của em. Em chỉ có thể xử lý đối với các trường hợp em đã liệt kê là bất kỳ số nào miễn chuyển sang thập phân và làm tròn bốn chữ số, số pi.

- Điểm tốt của đồ án của em là có thể sử dụng nhiều lần mà không cần khởi động lại. Các yêu cầu của đề đưa ra thì đã hoàn thành theo yêu cầu. Ngoài ra còn thêm phần trang trí giúp đồ án của em thêm phần nhiều sắc màu.
- Sau đây em sẽ chạy thử và chụp màn hình kết quả của bài toán tính tích phân của $f(x) = x^3 \sin(x)$ trong khoảng $[0; 1]$.



- **Lời kết:** Em xin cảm ơn thầy đã đọc hết bài báo cáo này của em. Trong quá trình làm và tiến hành làm báo cáo này em không thể tránh những sai sót trong lúc làm, nên em rất mong thầy có thể feedback trong Teams những lỗi về trình bày, lỗi thuật toán và lỗi code để em có thể khắc phục và hoàn thiện hơn trong tương lai ạ. Em mong nhận được nhận xét và phản hồi từ thầy, em xin cảm ơn thầy nhiều ạ.

-----HẾT-----