Problem 1.

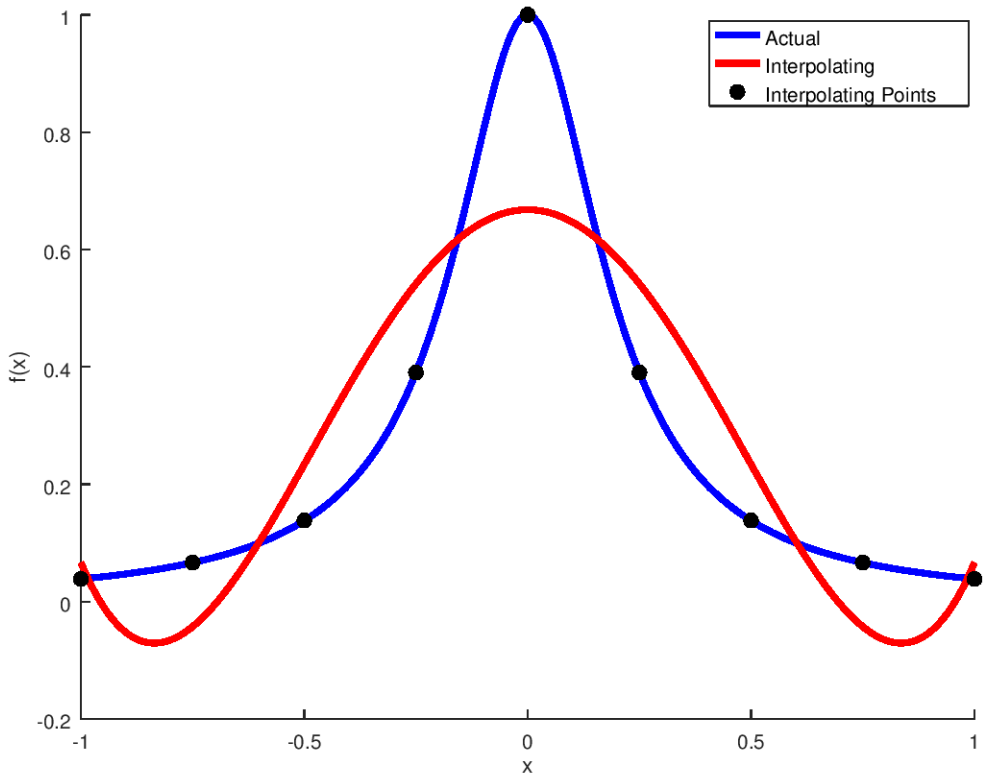
Use Matlab (or some other program) to calculate the polynomial which interpolates the function

$$f(x) = \frac{1}{1 + 25x^2} \tag{1}$$

at equally spaced points between -1 and 1, in steps of 0.25. You can use Matlab built-in to find the polynomial.

Plot the original function (in default color blue) and the interpolating polynomial (in red), and put circles around the interpolation points. Make sure you use enough points so that you get a smooth-looking curve.

Solution




```
function p1

clear all;
close all;
hold on;

```

```
f = @(x) 1 ./ (1 + 25 * x .^ 2);

% Actual curve.
_x = linspace(-1, 1, 200);
_y = f(_x);

% Interpolating points.
x = -1:0.25:1;
y = f(x);

% Fit a polynomial to the interpolating points.
p = polyval(polyfit(x, y, 5), _x);

plot(_x, _y, 'b', 'DisplayName', 'Actual', 'LineWidth', 3);
plot(_x,  p, 'r', 'DisplayName', 'Interpolating', 'LineWidth', 3);
scatter(x, y, 50, 'k', 'filled', 'DisplayName', 'Interpolating Points');
legend('show');
xlabel('x');
ylabel('f(x)');

end
```

Problem 2.

(a) Given a point x_0 and a step size h, use Lagrange polynomials to find the quadratic polynomial which interpolates a function f(x) at the points x_0-2h, x_0, and x_0+h.

(b) Integrate the polynomial on [x_0-2h, x_0+h] to produce an integration formula. Use the formula to estimate $\int_0^3 e^{-x}dx$.

Solution

Part (a)

The quadratic Lagrange polynomial P(t) is given by

$$\begin{aligned} P(t) = f(x_0 - 2h)\frac{(t-t_1)(t-t_2)}{(t_0-t_1)(t_0-t_2)} + f(x_0)\frac{(t-t_0)(t-t_2)}{(t_1-t_0)(t_1-t_2)} \\ + f(x_0+h)\frac{(t-t_0)(t-t_1)}{(t_2-t_0)(t_2-t_1)} \end{aligned}$$

In this case we have $t_0 = x_0 - 2h$, $t_1 = x_0$, and $t_2 = x_0 + h$.
Next we make the necessary substitutions and simplify the different parts of P(t).

$$\begin{aligned} (t-t_1)(t-t_) &= (t-x_0)(t-(x_0+h)) \\ (t-t_0)(t-t_2) &= (t-x_0+2h)(t-x_0-h) \\ (t-t_0)(t-t_1) &= (t-x_0+2h)(t-x0) \\ (t_0-t_1)(t_0-t_2) &= ((x_0-2h)-x_0)((x_0-2h)-(x_0+h)) \\ &= (-2h)(x_0-2h-x_0-h) \\ &= (-2h)(-3h) \\ &= 6h^2 \\ &= (t-x_0)(t-x_0-h) \\ (t_1-t_0)(t_1-t_2) &= (x_0-(x_0-2h))(x_0-(x_0+h)) \\ &= (2h)(-h) \\ &= -2h^2 \\ (t_2-t_0)(t_2-t_1) &= ((x_0+h)-(x_0-2h))((x_0-2h)-x0) \\ &= (3h)(h) \\ &= 3h^2 \end{aligned}$$

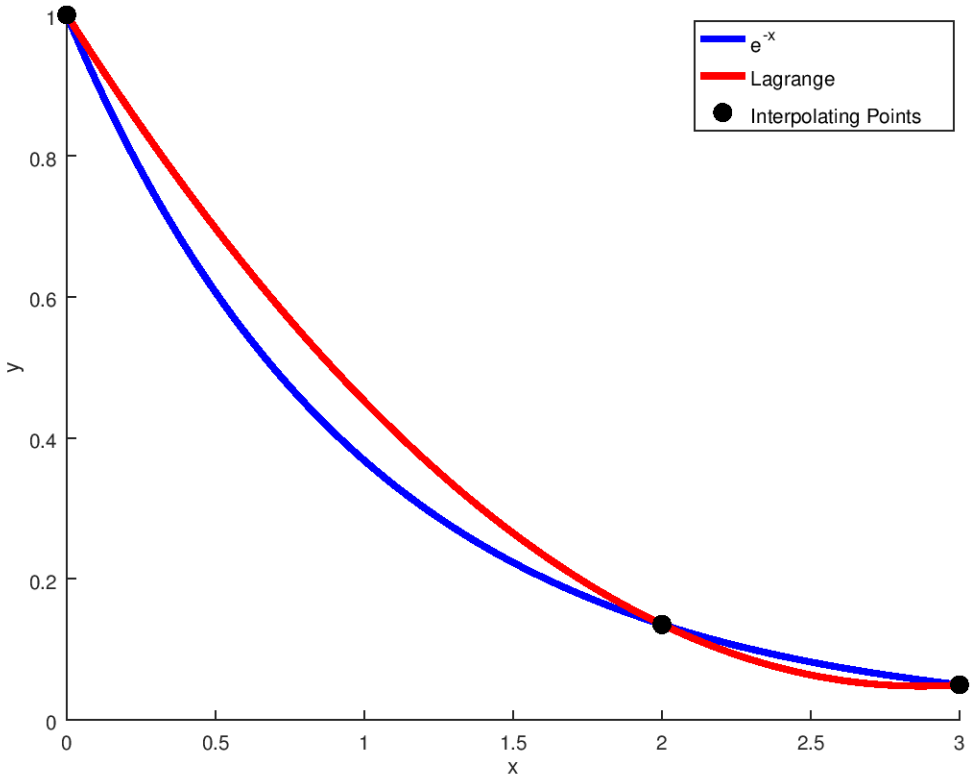
Thus we arrive at the following equation

$$\begin{aligned}P(t) &= \frac{f(x_0-2h)}{6h^2}(t-x_0)(t-x_0-h) + \frac{f(x_0)}{-2h^2}(t-x_0+2h)(t-x_0-h) \\ &+ \frac{f(x_0+h)}{3h^2}(t-x_0+2h)(t-x_0) \\ &= \frac{f(x_0-2h)}{6h^2}(t^2-(2x_0+h)t+(hx_0+x_0^2) + \frac{f(x_0)}{-2h^2}(t^2+(h-2x_0)t+(x_0^2-hx_0-2h^2) \\ &+ \frac{f(x_0+h)}{3h^2}(t^2+(2h-x_0)t+(x_0^2-2hx_0)\end{aligned}$$

Here we can see the that P(t) is quadratic with respect to t (as expected), solving for the coefficients of P(t) = At^2 + Bt + C produces

$$\begin{aligned}A &= f(x_0-2h)\frac{1}{6h^2} + f(x_0)\frac{1}{-2h^2} + f(x_0+h)\frac{1}{3h^2} \\ B &= f(x_0-2h)\frac{-2x_0-h}{6h^2} + f(x_0)\frac{h-2x_0}{-2h^2} + f(x_0+h)\frac{2h-2x_0}{3h^2} \\ C &= f(x_0-2h)\frac{hx_0+x_0^2}{6h^2} + f(x_0)\frac{x_0^2-hx_0-2h^2}{-2h^2} + f(x_0+h)\frac{x_0^2-2hx_0}{3h^2}\end{aligned}$$

Note that given h, x_0, and the interpolating points $f(x_0-2h)$, $f(x_0)$, and $f(x_0+h)$ each of the terms A, B, C are constant with respect to t.
Included below is a graph comparing $f(x) = e^{-x}$ with P(t) given $x_0 = 2$, h = 1.



Part (b)

$$\begin{aligned}\int_{x_0-2h}^{x_0+h} P(t) &= \int_{x_0-2h}^{x_0+h} At^2 + Bt + C dt \\ &= \left(\frac{1}{3}At^3 + \frac{1}{2}Bt^2 + Ct\right)\Big|_{x_0-2h}^{x_0+h} \\ &= \left(\frac{1}{3}A((x_0+h)^3 - (x_0-2h)^3) + \frac{1}{2}B((x_0+h)^2 - (x_0-2h)^2) + C((x_0+h) - (x_0-2h)\right)\end{aligned}$$

Expanding and simplifying our terms

$$\begin{aligned}(x_0+h)-(x_0-2h) &= x_0 - x_0 + h + 2h \\ &= 3h \\ (x_0+h)^2-(x_0-2h)^2 &= x_0^2 + 2x_0h + h^2 - x_0^2 + 4x_0h - 4h^2 \\ &= 6x_0h - 3h^2 \\ (x_0+h)^3-(x_0-2h)^3 &= x_0^3 + 3x_0^2h + 3x_0h^2 + h^3 \\ &\quad - x_0^3 + 6x_0^2h - 12x_0h^2 + 8h^3 \\ &= 9x_0^2 - 9x_0h^2 + 9h^3 \\ &= 9(x_0^2h - x_0h^2 + h^3)\end{aligned}$$

This gives us the following equation

$$\int_{x_0-2h}^{x_0+h} P(t) = 3hA(x_0^2 - x_0h + h^2) + \frac{1}{2}hB(6x_0 - 3h) + C3h$$

Using the same constant coefficients A, B, C that were derived in part (A).
We can use this formula to estimate $\int_0^3 e^{-x}dx$ by setting $x_0 = 2$, $h = 1$ and plugging those values into our equation for $\int_{x_0-2h}^{x_0+h} P(t)$.
By doing so we produce the following approximation:

$$\begin{aligned}\int_0^3 P(t)dt &= 1.0545 \\ &\approx \\ \int_0^3 e^{-x}dx &= 0.95021\end{aligned}$$

The estimate of the integral via the Lagrange polynomial has an error of 0.10429.

```
function p2

clear all;
close all;
hold on;

% Constants used for testing.
h = 1.0;
x0 = 2.0;

```

```
% Example function to test our interpolation method.
f = @(x) e .^ (-x);
_x = linspace(0, 3, 200);

% List of our interpolation points.
x = [x0 - 2*h, x0, x0 + h];

% Plot the target curve.
plot(_x, f(_x), 'b', 'DisplayName', 'e^{-x}', 'LineWidth', 3);

% Interpolating points.
F0 = f(x0 - 2*h);
F1 = f(x0);
F2 = f(x0 + h);

% These are the constant scalars of the quadratic formula.
A = (F0/(6*h^2)) + (F1/(-2*h^2)) + (F2/(3*h^2));
B = (F0*(-2*x0-h))/(6*h^2) + (F1*(h-2*x0))/(-2*h^2) +
 ↪  (F2*(2*h-2*x0))./(3*h^2);
C = (F0*(h*x0 + x0^2))/(6*h^2) + (F1*(x0^2-h*x0-2*h^2))/(-2*h^2) +
 ↪  (F2*(x0^2-2*h*x0))/(3*h^2);

% And this is the actual interpolating polynomial.
P = @(x) A*(x.^2) + B*x + C;
plot(_x, P(_x), 'r', 'DisplayName', 'Lagrange', 'LineWidth', 3);
scatter(x, f(x), 75, 'k', 'filled', 'DisplayName', 'Interpolating Points');

% Evaluate the integral of P(t) from (x0-2h) to (x0+h).
INT = 3*h*A*(x0^2-x0*h+h^2) + 0.5*h*B*(6*x0-3*h) + C*3*h % = 1.0545
% Built in numerical solution for the integral of f we are estimating.
quad(f, 0, 3) % = 0.95021

legend('show');
xlabel('x');
ylabel('y');

end
```

Problem 3.

For k=2, Iserles calls these the Nystrom and Milne methods. That is, start with

$$y(t_{n+2}) = y(t_n) + \int_{t_n}^{t_{n+2}} f(\tau, y(\tau))d\tau, \tag{2}$$

and interpolating at t_n, t_{n+1} (explicit) or t_n, t_{n+1}, t_{n+2} (implicit).
Find the leading error term, and the order of the method, for both cases.

Solution

Part (a)

Given $y(t_{n+2}) = y(t_n) + \int_{t_n}^{t_{n+2}} f(\tau, y(\tau))d\tau$, to solve for the explicit case (and thus find the error), we must approximate $\int_{t_n}^{t_{n+2}} f(\tau, y(\tau))d\tau$ using a Lagrange polynomial interpolating the points $f(t_n, y_n)$ and $f(t_{n+1}, y(t_{n+1}))$.
That is $y(t_{n+2}) = y(t_n) + \int_{t_n}^{t_{n+2}} f(\tau, y(\tau))d\tau \approx y(t_n) + \int_{t_n}^{t_{n+2}} P(\tau)d\tau$. Where P(t) is the Lagrange polynomial.

$$\begin{aligned} P(t) &= f(t_n, y_n)\frac{t - t_{n+1}}{t_n - t_{n+1}} + f(t_{n+1}, y(t_{n+1}))\frac{t - t_n}{t_{n+1} - t_n} \\ &= \frac{f(t_n, y_n)}{-h}(t - t_{n+1}) + \frac{f(t_{n+1}, y(t_{n+1}))}{h}(t - t_n) \end{aligned}$$

Next we integrate each side from t_n to t_{n+2} with respect to t

$$\begin{aligned} \int_{t_n}^{t_{n+2}} P(\tau)d\tau &= \frac{f(t_n, y_n)}{-h}\int_{t_n}^{t_{n+2}} (\tau - t_{n+1})d\tau + \frac{f(t_{n+1}, y(t_{n+1}))}{h}\int_{t_n}^{t_{n+2}} (\tau - t_n)d\tau \\ &= \frac{f(t_n, y_n)}{-h}(\frac{1}{2}\tau^2 - (t_{n+1})\tau)\Big|_{t_n}^{t_{n+2}} + \frac{f(t_{n+1}, y(t_{n+1}))}{h}(\frac{1}{2}\tau^2 - (t_n)\tau)\Big|_{t_n}^{t_{n+2}} \end{aligned}$$

Evaluating the integrals...

$$\begin{aligned}
(\frac{1}{2}\tau^2 - (t_{n+1})\tau)\Big|_{t_n}^{t_{n+2}} &= \frac{1}{2}(t_{n+2}^2) - (t_{n+1})(t_{n+2}) - \frac{1}{2}t_n^2 + (t_{n+1})t_n \\
&= \frac{1}{2}(t_{n+2}^2 - t_n^2) + (t_{n+1})(t_n - t_{n+2}) \\
&= \frac{1}{2}(t_{n+2} + t_n)(t_{n+2} - t_n) + -2h(t_{n+1}) \\
&= h(t_{n+2} + t_n) + -2h(t_{n+1}) \\
&= h(t_{n+2} + t_n - t_{n+1} - t_{n+1}) \\
&= h(t_{n+2} - t_{n+1} + t_n - t_{n+1}) \\
&= h(h - h) \\
&= 0 \\
(\frac{1}{2}\tau^2 - (t_n)\tau)\Big|_{t_n}^{t_{n+2}} &= \frac{1}{2}(t_{n+2})^2 - (t_n)(t_{n+2}) - \frac{1}{2}t_n^2 - (t_{n+1})(t_{n+2}) \\
&= \frac{1}{2}(t_{n+2}^2 - t_n^2) + (t_n)(t_n - t_{n+2}) \\
&= \frac{1}{2}(t_{n+2} + t_n)(t_{n+2} - t_n) + -2h(t_n) \\
&= h(t_{n+2} + t_n) + -2h(t_n) \\
&= h(t_{n+2} + t_n + -t_n - t_n) \\
&= h(t_{n+2} - t_n) \\
&= h(2h)
\end{aligned}$$

Substituting these results into our original equation

$$\begin{aligned}
\int_{t_n}^{t_{n+2}} P(\tau)d\tau &= \frac{f(t_n, y_n)}{-h}0 + \frac{f(t_{n+1}, y(t_{n+1}))}{h}h(2h) \\
&= 2hf(t_{n+1}, y(t_{n+1}))
\end{aligned}$$

From

$$y(t_{n+2}) = y(t_n) + \int_{t_n}^{t_{n+2}} f(\tau, y(\tau))d\tau \approx y(t_n) + \int_{t_n}^{t_{n+2}} P(\tau)d\tau = y(t_n) + 2hf(t_{n+1}, y(t_{n+1}))$$

We get our k-step method as follows

$$y_{n+2} = y_n + 2hf(t_{n+1}, y(t_{n+1}))$$

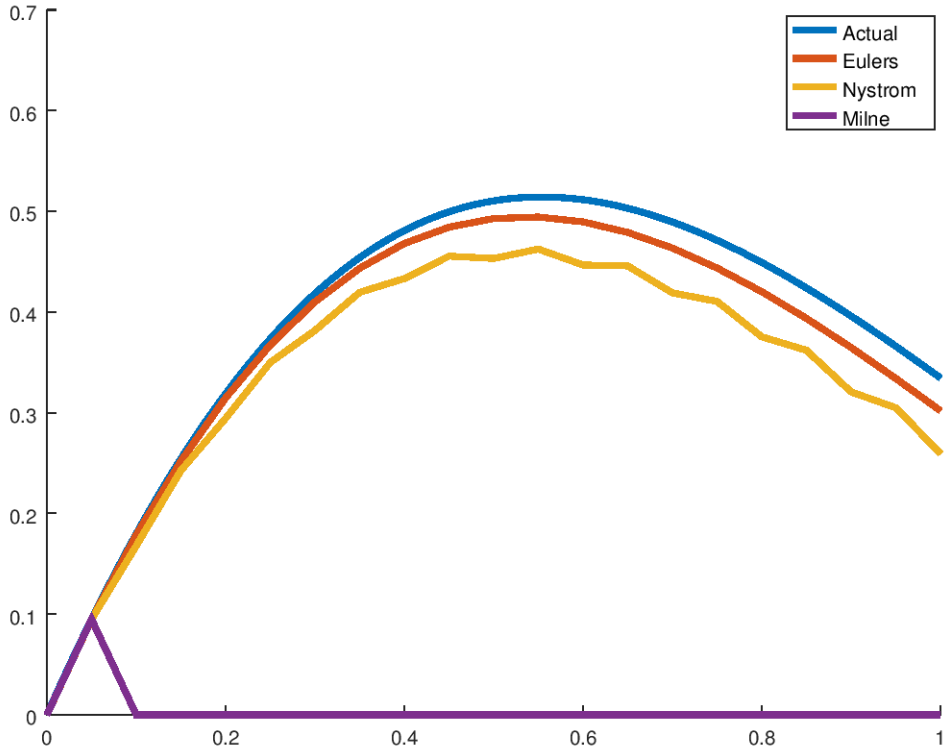
The coefficients for the general k-step then are $a_1 = -1, a_1 = 0, b_2 = 0, b_1 = 2, b_0 = 0$. Thus $p(w) = w^2 - 1$ and $\sigma(w) = 2w$.

$$\begin{aligned} p(e^h) - h\sigma(e^h) &= e^{2h} - 1 - h2e^h \\ &= \frac{1}{3}h^3 + \frac{1}{3}h^4 + \frac{11}{60}h^5 + \dots \\ &= \frac{1}{3}h^3 + \mathrm{O}(h^4) \end{aligned}$$

This implies that the Nystrom method has order 2, with a leading error term $\frac{1}{3}h^3$.

Part (b)

Euler vs Nystrom vs Milne for 20 steps.




```
function p3

clear all;
close all;
hold on;

y  = @(t) (e .^ -t) .* (sin(2 .* t));
```

```
dy = @(t, y) -y + (2 .* e .^ -t) .* cos(2 .* t);

% Plot the function we are trying to numerically compute.
_x = linspace(0, 1, 200);
_y = y(_x);
plot(_x, y(_x), 'DisplayName', 'Actual', 'LineWidth', 3);

% Parameters
start = 0;
stop = 1;
steps = 20;
h = (stop - start)/steps;

% IVP
x = start:h:stop;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%        Euler Method.                             %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

yk = zeros(steps + 1, 1);
yk(1) = y(0);

% Run Eulers method.
for k = 1:steps
    tk = start + h * k;
    yk(k+1) = yk(k) + h * dy(tk, yk(k));
end

plot(x, yk, 'DisplayName', 'Euler', 'LineWidth', 3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%        Nystrom (explicit method).                %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

yk = zeros(steps + 1, 1);
yk(1) = y(0);
yk(2) = y(h);

% Run the Nystrom method.
for k = 1:(steps-1)
    tknext = start + h * (k+1);
    yk(k+2) = yk(k) + 2 * h * dy(tknext, yk(k+1));
end
```

```
52
53 plot(x, yk, 'DisplayName', 'Nystrom', 'LineWidth', 3);
54
55 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
56 %%         Milne (implicit method).          %%
57 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
58
59 yk = zeros(steps + 1, 1);
60 yk(1) = y(0);
61 yk(2) = y(h);
62
63 % Run the Milne method.
64 for k = 1:(steps-1)
65     % TODO
66 end
67
68 plot(x, yk, 'DisplayName', 'Milne', 'LineWidth', 3);
69
70 legend('show');
71
72 end
```

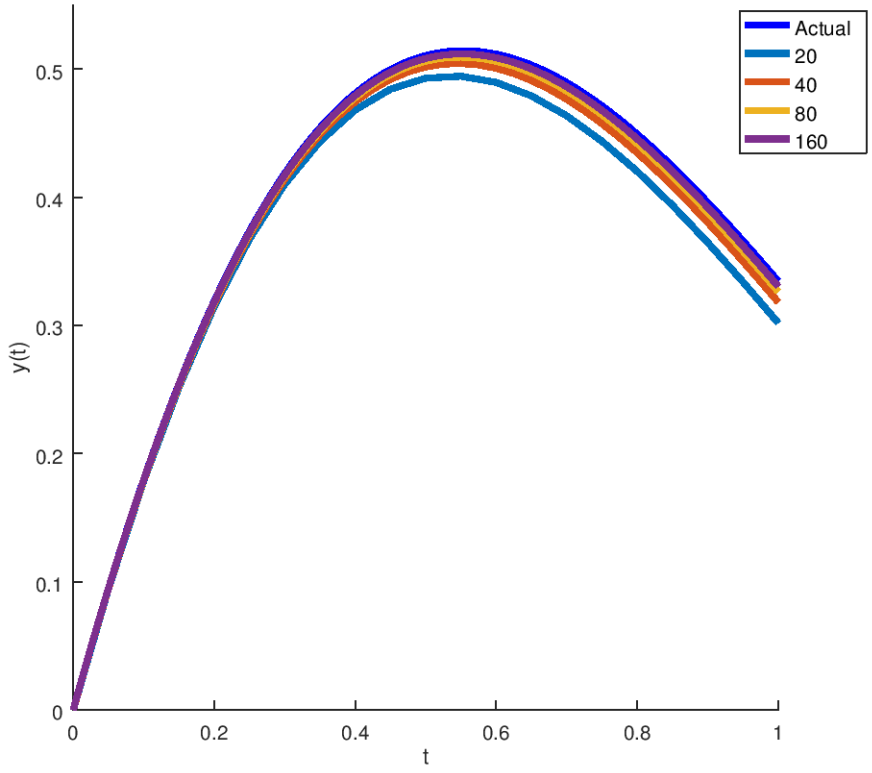
Problem 4.

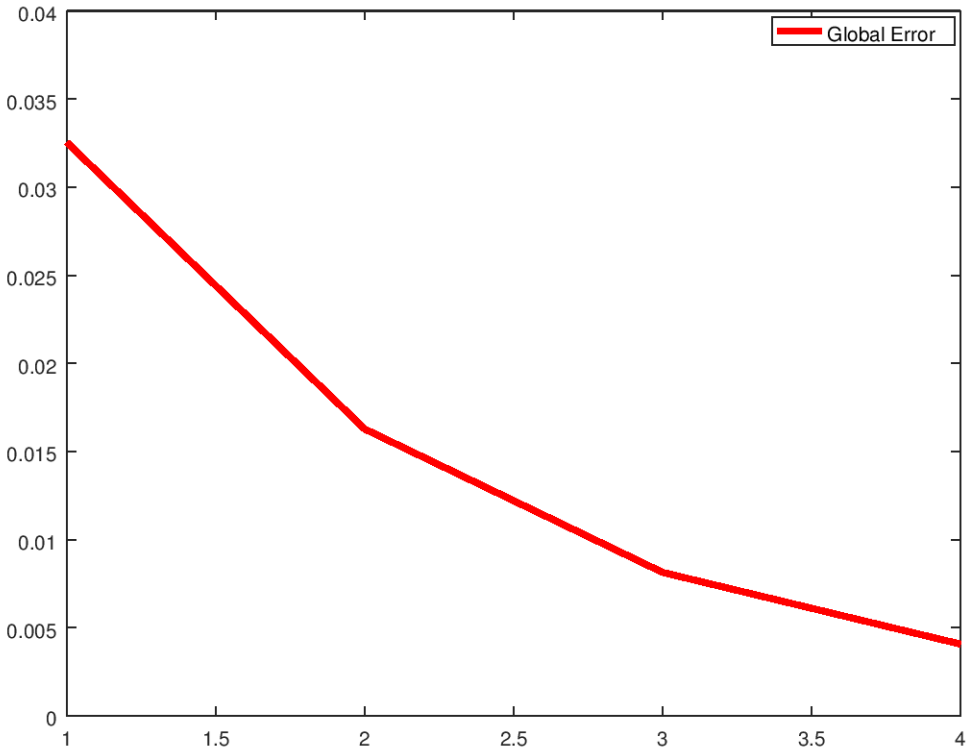
The IVP

$$y' = -y + e^{-t}\cos(2t) \tag{3}$$
$$y(0) = 0 \tag{4}$$

has the true solution $y(t) = e^{-t}\sin(2t)$. Solve it numerically, using Euler's method with 20, 40, 80, 160 steps from 0 to 1, and compute the global error at the endpoint for each stepsize. Verify that the error goes down approximately linearly.

Solution





Note that the error term (graphed above) descends approximately linearly with respect to the step size.

```
function p4

clear all;
close all;
hold on;
axis([0, 1, 0, 0.55], 'square');

y  = @(t) (e .^ -t) .* (sin(2 .* t));
dy = @(t, y) -y + (2 .* e .^ -t) .* cos(2 .* t);

_x = linspace(0, 1, 200);
_y = y(_x);
plot(_x, y(_x), 'DisplayName', 'Actual', 'b', 'LineWidth', 3);

% Where to start and stop the approximation.
start = 0;
stop = 1;

ERR = zeros(4, 1);
```

```
steps = 20 .* 2 .^ (0:3);

for idx = 1:4

    h = (stop - start)/steps(idx);
    yk = zeros(steps + 1, 1);
    x = start:h:stop;

    yk(1) = 0; % Given initial point.
    for k = 1:steps(idx)
        tk = start + h * k;
        yk(k+1) = yk(k) + h * dy(tk, yk(k));
    end

    plot(x, yk, 'DisplayName', num2str(steps(idx)), 'LineWidth', 3);
    ERR(idx) = norm(yk - y(start:h:stop)', inf);
end

xlabel('t');
ylabel('y(t)');
legend('show');

%%%%%%%%%%%%%%%%%%%%%%%%
%% Graph the error. %%
%%%%%%%%%%%%%%%%%%%%%%%%
hold;
plot(1:4, ERR, 'r', 'LineWidth', 3, 'DisplayName', 'Global Error');
legend('show');

end
```

Problem 5.

Solve

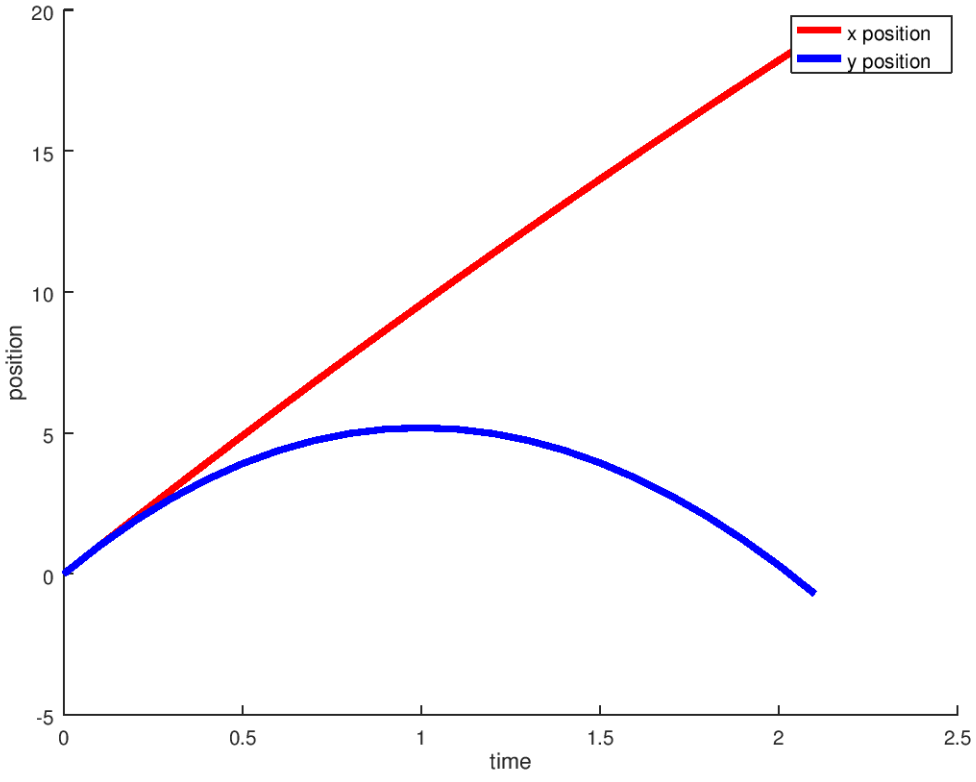
$$y''(t) = \begin{bmatrix} 0 \\ -10 \end{bmatrix} - 0.1y'(t),$$

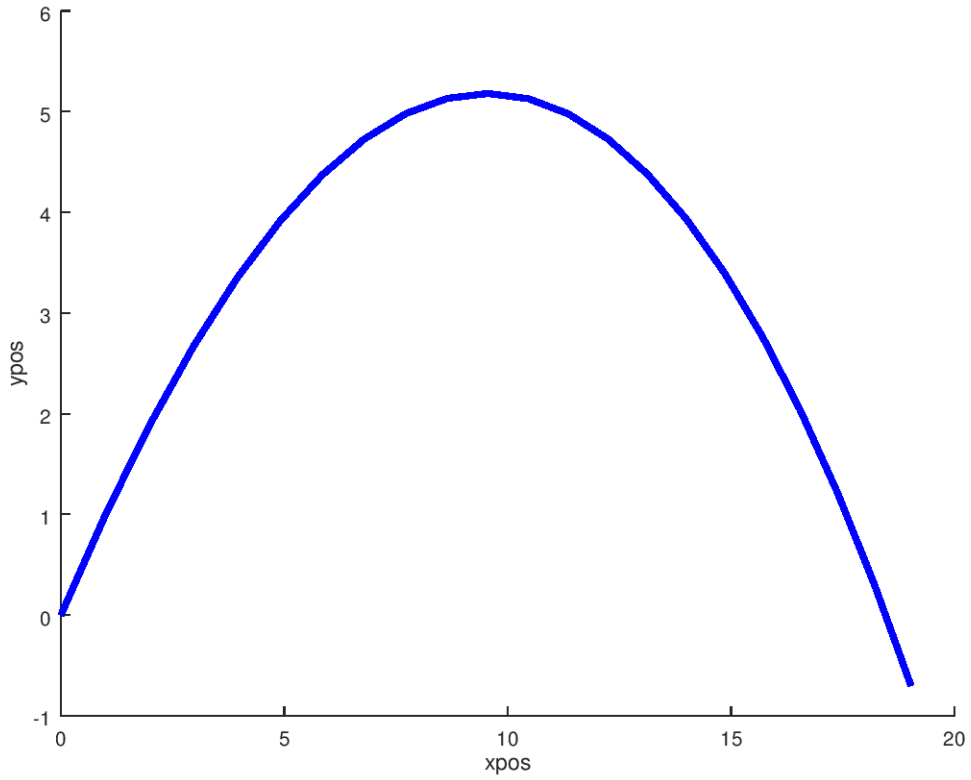
$$y(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

$$y'(0) = \begin{bmatrix} 10 \\ 10 \end{bmatrix}$$

by using Euler's method with stepsize h = 0.1, until y_2 becomes 0 or negative.
Plot y_1 and $y_)$ as functions of t (horizontal and vertical distance as functions of time), and also plot y_2 versus y_1. That will show the path of the object. It should look a bit like a parabola, but compressed on the right because of friction.

Solution






```
function p5

clear all;
close all;
hold on;

% Numerical solution initialized by the given initial values.
y1 = [[0; 0]];                % y1 = y
y2 = [[10; 10]];        % y2 = y'

dy2 = @(t, y2) [0; -10] - (0.1 .* y2);
h = 0.1;         % Constant step size.
_t = [0];         % List of time steps for plotting.

% Need at least one pass to get off the ground...
do
    % Get the current time step.
    t = _t(length(_t));

    % Previous elements we are stepping off from.
    prev_y2 = y2(:, length( y2(2,:) ));
```

```
    prev_y1 = y1(:, length( y1(2,:) ));

    % Step the simulation.
    y1 = [y1, prev_y1 + h .* prev_y2];
    y2 = [y2, prev_y2 + h .* dy2(t, prev_y2)];

    % Step our method forward.
    _t = [_t, t+h];

% Until we reach the ground again.
until y1(2, length(y1(1,:))) <= 0 % Octave not Matlab :)

% Plot horizontal and vertical position with respect to to time.
xlabel('time');
ylabel('position');
xplot = plot(_t, y1(1,:), 'LineWidth', 3, 'r');
yplot = plot(_t, y1(2,:), 'LineWidth', 3, 'b');
legend('x position', 'y position');

% Plot horizontal against vertical position.
hold;
xlabel('xpos');
ylabel('ypos');
plot(y1(1,:), y1(2,:), 'LineWidth', 3, 'b');

end
```