Probleem 1:

(a)

Vir elke stap word 2 vermenigvuldigings en 1 optelling uitgevoer. Daar is n stappe, dus word $2n$ vermenigvuldigings en n optellings in totaal uitgevoer.

(b)

Vir elke stap word 1 vermenigvuldiging en 1 optelling uitgevoer. Daar is n stappe, dus word n vermenigvuldigings en n optellings in totaal uitgevoer.

Die algoritme van deel (b) voer dus n vermenigvuldigings minder uit as die algortime van deel (a).

(c)

```
function p = horner(a,x)

% MATLAB programmetjie wat Horner se algoritme gebruik om
%  p(x) = a(1) + a(2)*x + ... + a(n+1)*x^n  te evalueer.

n = length(a) - 1;
u = a(n+1);

for k = n-1:-1:0,
    u = a(k+1) + x*u;
end

p = u;
```

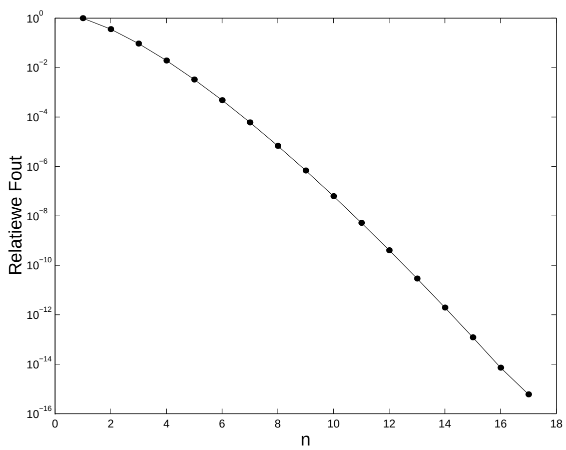
Om hierdie program te toets, kan MATLAB se `polyval` gebruik word. Let op, `polyval` se koëffisiënte word andersom genommer; ons moet dus `p1 = horner(a,x)` met `p2 = polyval(fliplr(a),x)` vergelyk.

Probleem 2:

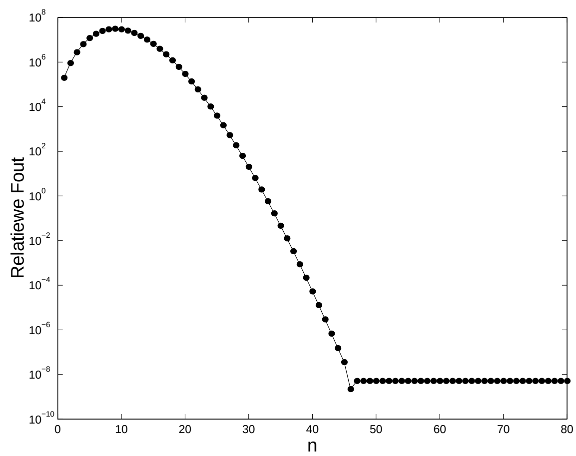
(a) en (b)

Die MATLAB-kode, en die grafieke wat verkry word:

```
x = -1;                     % deel (b):    x = -10;
eksak = exp(x);

RE = [];

for n = 1:20,               % deel (b):    for n = 1:80,
    a = 1./gamma(1:n+1);
    p = polyval(fliplr(a),x);
    RE = [RE; abs((p-eksak)/eksak)];
end

semilogy(RE);
hold on
semilogy(RE,'.','MarkerSize',16);
xlabel('n','FontSize',16); ylabel('Relatiewe Fout','FontSize',16);
```



(a)



(b)

In (a) word volle akkuraatheid bereik, d.w.s. Rel.Fout $\approx 10^{-16}$. Volle akkuraatheid word egter nie in (b) bereik nie, waar Rel.Fout nie kleiner as ongeveer 5×10^{-9} word nie.

Die verklaring is weer eens kansellasiefoute: die individuele terme in deel (b) is groot en van afwisselende teken, bv. terme 9 tot 12 in die polinoom lyk soos volg:

```
2.4802e+003   -2.7557e+003    2.7557e+003   -2.5052e+003
```

Wanneer terme so groot soos hierdie bymekaar getel word om 'n uiteindelike resultaat so klein as

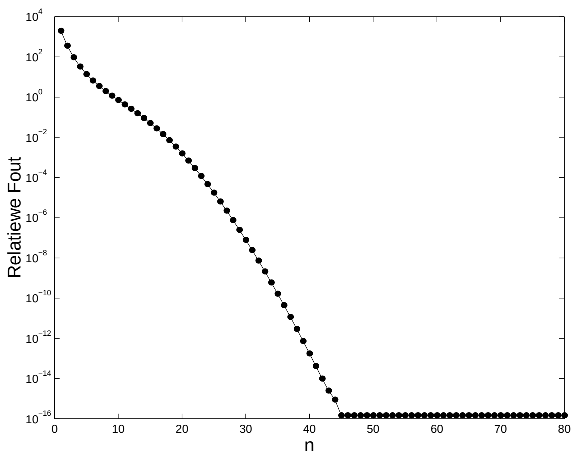
```
exp(-10) = 4.5400e-005
```

te lewer, moes kansellasiefoute iewers langs die pad ingesluip het.

(c)

Die MATLAB-kode (die plot-instruksies van deel (b) bly onveranderd):

```
x = 10;
eksak = exp(-x);
RE = [];
for n = 1:80,
    a = 1./gamma(1:n+1);
    p = polyval(fliplr(a),x);
    p = 1/p;
    RE = [RE; abs((p-eksak)/eksak)];
end
```



Met hierdie algoritme is volle akkuraatheid bereik. Hierdie polinoom bestaan uit positiewe terme (d.w.s. geen aftrekkings nie) en die kansellasie van deel (b) vind nie plaas nie.

Probleem 3:

(a)

e^{-t} is 'n streng-afnemende funksie op $(0,1)$, dus

$$\begin{aligned} e^{-1} &< e^{-t} < 1 \\ t^n e^{-1} &< t^n e^{-t} < t^n \\ \int_0^1 t^n e^{-1} \mathrm{d}t &< \int_0^1 t^n e^{-t} \mathrm{d}t < \int_0^1 t^n \mathrm{d}t \\ \frac{1}{e(1+n)} &< y_n < \frac{1}{1+n}. \end{aligned}$$

Nou, $\frac{1}{e(1+n)} \to 0$ en $\frac{1}{1+n} \to 0$ as $n \to \infty$. Dus, $y_n \to 0$ as $n \to \infty$.

(b)

Vir $n \geq 1$,

$$\begin{aligned} y_n &= \int_0^1 t^n e^{-t} \mathrm{d}t \\ &= -\int_0^1 t^n \frac{\mathrm{d}}{\mathrm{d}t} e^{-t} \mathrm{d}t \\ &= -t^n e^{-t}\Big|_0^1 + \int_0^1 n t^{n-1} e^{-t} \mathrm{d}t \\ &= -e^{-1} + n \int_0^1 t^{n-1} e^{-t} \mathrm{d}t \\ &= n y_{n-1} - 1/e. \end{aligned}$$

Vir $n = 0$,

$$y_0 = \int_0^1 e^{-t} \mathrm{d}t = 1 - 1/e.$$

(c)

Die MATLAB-kode:

```
e = exp(1);
y = 1 - 1/e; Y = y;
for n = 1:25,
    y = n*y - 1/e; Y = [Y;y];
end
```

n	y_n
0	6.321205588285577e-001
1	2.642411176571154e-001
2	1.606027941427885e-001
3	1.139289412569232e-001
4	8.783632385625056e-002
5	7.130217810981054e-002
6	5.993362748742098e-002
7	5.165595124050459e-002
8	4.536816875259447e-002
9	4.043407760190798e-002
10	3.646133484763753e-002
11	3.319524215257053e-002
12	3.046346465940403e-002
13	2.814559940081007e-002
14	2.615895043989874e-002
15	2.450481542703886e-002
16	2.419760566117951e-002
17	4.347985506860941e-002
18	4.147579500635271e-001
19	7.512521610035572e+000
20	1.498825527595400e+002
21	3.147165728509168e+003
22	6.923727814776053e+004
23	1.592457029519051e+006
24	3.821896834057778e+007
25	9.554742081465652e+008

Die numeriese resultate is nie in ooreenstemming met die teorie nie. Teoreties moet y_n na nul streef soos n toeneem, maar numeries lyk dit of y_n onbegrens groot word.

(d)

Versteur y_0 na $y_0 + \epsilon$, en dui die nuwe ry van getalle aan met $\{\widetilde{y}_n\}$. Dan

$$\begin{aligned} \widetilde{y}_0 &= y_0 + \epsilon \\ \widetilde{y}_n &= n\widetilde{y}_{n-1} - 1/e. \end{aligned} \tag{1}$$

Die oorspronklike ry van getalle bevredig

$$y_n = ny_{n-1} - 1/e. \tag{2}$$

Tref (2) van (1) af,

$$\begin{aligned} \tilde{y}_n - y_n &= n(\tilde{y}_{n-1} - y_{n-1}) \\ e_n &= ne_{n-1} \end{aligned}$$

waar $e_n = \tilde{y}_n - y_n$ die absolute fout op stap n voorstel. Nou

$$e_0 = \epsilon, \quad e_1 = 1 \cdot e_0 = \epsilon, \quad e_2 = 2 \cdot e_1 = 2\epsilon, \quad e_3 = 3 \cdot e_2 = 6\epsilon, \quad \text{ens.}$$

Dus $e_n = n!\epsilon$, wat aantoon dat die fout onbegrens groei as $n \to \infty$.

(e)

Die terugwaartse rekursie formule word gegee deur

$$y_{n-1} = \frac{y_n + 1/e}{n}.$$

Die MATLAB-kode vir hierdie rekursie formule:

```
e = exp(1);
y = 0; Y = y;
for n = 40:-1:1,
    y = (y + 1/e)/n; Y = [Y;y];
end
```

```
 n             y_n                      y_n (eksak)
----------------------------------------------------
40             0
39   9.196986029286057e-003
38   9.668626338480214e-003
37   9.935475460787433e-003
36   1.021121396303324e-002
35   1.050251819817988e-002
34   1.081091312484635e-002
33   1.113795159694967e-002
32   1.148537553843612e-002
31   1.185515052218370e-002
30   1.224950295785890e-002
29   1.267096480431004e-002
28   1.312242779226732e-002
27   1.360720960584677e-002
26   1.412913521397367e-002
```

25	1.469263755328523e-002	
24	1.530288314898910e-002	
23	1.596593018001797e-002	
22	1.668892918919392e-002	
21	1.748038047093801e-002	
20	1.835046769725621e-002	1.835046769725622e-002
19	1.931149544343492e-002	
18	2.037847034815143e-002	
17	2.156988397331076e-002	
16	2.290878383204430e-002	
15	2.442426406271791e-002	2.442426406271792e-002
14	2.615358034894401e-002	
13	2.814521582288473e-002	
12	3.046343515340977e-002	
11	3.319523969373767e-002	
10	3.646133462410727e-002	3.646133462410727e-002
9	4.043407757955496e-002	
8	4.536816875011081e-002	
7	5.165595124019413e-002	
6	5.993362748737663e-002	
5	7.130217810980315e-002	7.130217810980316e-002
4	8.783632385624909e-002	
3	1.139289412569229e-001	
2	1.606027941427884e-001	
1	2.642411176571153e-001	
0	6.321205588285577e-001	6.321205588285578e-001

Die eksakte waardes van y_n (bereken m.b.v. Mathematica) bevestig dat die terugwaartse rekursie volle akkuraatheid lewer. Die volgende Mathematica instruksies kan gebruik word om tot 16 syfers die integraal akkuraat te bereken:

```
n = 20;
N[NIntegrate[t^n Exp[-t],{t,0,1}],16]
```