Írjunk scriptet `gsz.m` néven, amely kiszámolja

$$S_n = \sum_{i=1}^{n} \frac{1}{i(i+1)} = \frac{1}{1 \cdot 2} + \frac{1}{2 \cdot 3} + \ldots + \frac{1}{n \cdot (n+1)}$$

értékét mindkét algoritmussal $n = 999999$ esetén. Melyik lesz pontosabb? A hibák kiszámításánál viszonyítsunk az $S_n = 1 - \frac{1}{n+1}$ képlettel számított értékhez, ugyanis ez utóbbi képlet zárt alakban megadja a fenti összeget.

`gsz.m`

```
%Szamoljuk ki az S_n = 1/(1*2) + 1/(2*3) + ... + 1/(n(n+1)) osszeget.
n = 999999;
%az osszeg zart alakban is megadhato:
S_n = 1 - 1/(n+1);
%eloszor abban a sorrendben adjuk ossze a szamokat, ahogy jonnek
s1 = 0;
for i=1:n
     s1 = s1 + 1/(i*(i+1));
end
%azutan forditott sorrendben, a vegetol haladva visszafele
s2 = 0;
for i=n:-1:1
     s2 = s2 + 1/(i*(i+1));
end
elteres1 = abs(S_n-s1)
elteres2 = abs(S_n-s2)
```

Legyen most $n \in \mathbb{N}$, és

$$I_n = \int_0^1 \frac{x^n}{10 + x} \mathrm{d}x.$$

Ekkor I_n kiszámítható a következő rekurzióval:

$$I_n = \frac{1}{n} - 10 I_{n-1},$$

amely azonban numerikusan instabil algoritmust eredményez. Írjunk programot ennek a rekurziónak az elvégzésére `gsz_integral1.m` néven.

gsz_integral1.m

```
function I = gsz_integral1(n)
I = zeros(n+1, 1);
I(1) = log(1.1);
% I(1) = I_0, es ehhez hasonloan, az I tomb k-adik helyen I_{k-1} all.

for k = 1:n
     I(k+1) = 1/k - 10*I(k);
end
```

Próbáljuk ki n kis értékeire, majd $n = 19$, $n = 20$-ra is.

A fenti programba az `I = zeros(n+1, 1);` sor elhagyható, a szerepe az, hogy az `I` vektornak előzetesen lefoglalnuk egy megfelelő méretű helyet.

Vegyük észre, hogy a fenti rekurziós összefüggésből egyszerű átrendezéssel könnyen gyártható egy másik rekurzió is, amely visszafelé lépeget:

$$I_{n-1} = 0.1(\frac{1}{n} - I_n)$$

A kezdőérték egy n-nél nagyobb értékhez tartozó integrál, amely már gyakorlatilag nullának vehető (I_n ugyanis tart a nullához). Az alábbi programba ezt a stabil algoritmust is beépítettük. Ebben a programban tetszőleges n esetén $I_{n+30} \approx 0$-ból indítjuk a rekurziót.

gsz_integral.m

```
function Integral = gsz_integral(n)
%egy numerikusan instabil algoritmus:  integralszamitas rekurzioval
%Az I_n = int_0^1 x^n/(10+x)dx integral kiszamitasahoz egy rekurziot
%hasznalunk, a kapott algoritmus azonban numerikusan instabil.

I = zeros(n+1, 1);
I(1) = log(1.1);
% I(1) = I_0, es ehhez hasonloan, az I tomb k-adik helyen I_{k-1} all.

for k = 1:n
```

```
     I(k+1) = 1/k - 10*I(k);
end

%Stabil algoritmussal:
N = n + 30;
J = zeros(N+1, 1);
%J(end) = 0;

for k = N:-1:1
     J(k) = 0.1*(1/k - J(k+1));
end

%Az Integral vektor elso oszlopa az instabil algoritmussal kapott eredmeny,
%a masodik a stabil algoritmussal kapott eredmeny.
Integral = zeros(n+1, 2);
Integral(:, 1) = I;
Integral(:, 2) = J(1:n+1);
```

Időmérés a Matlabban

A Matlabban egyszerűen tudunk futási időt mérni: a kiválasztott programrész elé szúrjuk be a `tic`, mögé pedig a `toc` parancsot. Erre a Matlab kiírja a programrész futási idejét másodpercekben. Ha a futási időt pl. az `ido` nevű változóba szeretnénk elmenteni, akkor `toc` helyett `ido = toc` írható.