```
> restart:
> Digits:=40:
```

3-connected->2-connected

Expression of M the generating function of rooted 3-connected planar maps

```
> Exp_M:=x^2*y^2*(1/(1+x*y)+1/(1+y)-1-(1+u)^2*(1+v)^2/(1+u+v)
  ^3);
```

$$Exp\_M := x^2 y^2 \left(\frac{1}{x y + 1} + \frac{1}{1 + y} - 1 - \frac{(1+u)^2 (1+v)^2}{(1+u+v)^3} \right) \quad \textbf{(1.1.1)}$$

Expressions of generating functions of bicolored binary trees

We obtain expressions of the generating functions of bicolored binary trees and their partial derivatives. As they will be evaluated at (x,D) in the systems given after, we replace y by D. These expressions will be placed in the systems verified by networks, pointed networks, and bi-pointed networks. Here u and v stand respectively for the generating functions of black-rooted and white-rooted bicolored binary trees.

```
> system_u_v:={u=x*D*(1+v)^2,v=D*(1+u)^2};
```

$$system\_u\_v := \{u = x\,D\,(1+v)^2,\ v = D\,(1+u)^2\} \quad \textbf{(1.2.1)}$$

```
> Equation_dxu_dxv:=subs({diff(u(x,D),x)=dxu,diff(v(x,D),x)=
  dxv,u(x,D)=u,v(x,D)=v},diff(subs(u=u(x,D),v=v(x,D),
  system_u_v),x)):
  solution_dxu_dxv_u_v:=solve(Equation_dxu_dxv,{dxu,dxv}):
  Equation_dyu_dyv:=subs({diff(u(x,D),D)=dyu,diff(v(x,D),D)=
  dyv,u(x,D)=u,v(x,D)=v},diff(subs(u=u(x,D),v=v(x,D),
  system_u_v),D)):
  solution_dyu_dyv_u_v:=solve(Equation_dyu_dyv,{dyu,dyv}):
  system_derivate_binary_tree:={
     dxu=subs(solution_dxu_dxv_u_v,dxu),
     dxv=subs(solution_dxu_dxv_u_v,dxv),
     dyu=subs(solution_dyu_dyv_u_v,dyu),
     dyv=subs(solution_dyu_dyv_u_v,dyv)
  };
```

$$system\_derivate\_binary\_tree := \left\{ dxu = -\frac{D\left(v^2+2v+1\right)}{4D^2uvx+4D^2ux+4D^2vx+4D^2x-1},\ dxv = -\frac{2D^2(1+u)\left(v^2+2v+1\right)}{4D^2uvx+4D^2ux+4D^2vx+4D^2x-1},\ dyu = -\frac{x\left(2Du^2v+2Du^2+4Duv+4Du+2Dv+v^2+2D+2v+1\right)}{4D^2uvx+4D^2ux+4D^2vx+4D^2x-1},\ dyv = -\frac{2Duv^2x+4Duvx+2Dv^2x+2Dux+4Dvx+2Dx+u^2+2u+1}{4D^2uvx+4D^2ux+4D^2vx+4D^2x-1} \right\} \quad \textbf{(1.2.2)}$$

```
> system_bi_derivate_binary_tree:={
```

```
   dxxu=factor(subs(system_derivate_binary_tree,subs({diff(u
(x,D),x)=dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs(
{u=u(x,D),v=v(x,D)},subs(system_derivate_binary_tree,dxu)),x)
))),
   dxxv=factor(subs(system_derivate_binary_tree,subs({diff(u
(x,D),x)=dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs(
{u=u(x,D),v=v(x,D)},subs(system_derivate_binary_tree,dxv)),x)
))),
   dxyu=factor(subs(system_derivate_binary_tree,subs({diff(u
(x,D),x)=dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs(
{u=u(x,D),v=v(x,D)},subs(system_derivate_binary_tree,dyu)),x)
))),
   dxyv=factor(subs(system_derivate_binary_tree,subs({diff(u
(x,D),x)=dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs(
{u=u(x,D),v=v(x,D)},subs(system_derivate_binary_tree,dyv)),x)
))),
   dyyu=factor(subs(system_derivate_binary_tree,subs({diff(u
(x,D),D)=dyu,diff(v(x,D),D)=dyv,u(x,D)=u,v(x,D)=v},diff(subs(
{u=u(x,D),v=v(x,D)},subs(system_derivate_binary_tree,dyu)),D)
))),
   dyyv=factor(subs(system_derivate_binary_tree,subs({diff(u
(x,D),D)=dyu,diff(v(x,D),D)=dyv,u(x,D)=u,v(x,D)=v},diff(subs(
{u=u(x,D),v=v(x,D)},subs(system_derivate_binary_tree,dyv)),D)
)))
   }:
> system_tri_derivate_binary_tree:={

   dxxxu=factor(subs(system_derivate_binary_tree,subs({diff(u(x,
   D),x)=dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs({u=
   u(x,D),v=v(x,D)},subs(system_bi_derivate_binary_tree,dxxu)),
   x)))),
   dxxxv=factor(subs(system_derivate_binary_tree,subs({diff(u(x,
   D),x)=dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs({u=
   u(x,D),v=v(x,D)},subs(system_bi_derivate_binary_tree,dxxv)),
   x)))),
   dxxyu=factor(subs(system_derivate_binary_tree,subs({diff(u(x,
   D),x)=dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs({u=
   u(x,D),v=v(x,D)},subs(system_bi_derivate_binary_tree,dxyu)),
   x)))),
   dxxyv=factor(subs(system_derivate_binary_tree,subs({diff(u(x,
   D),x)=dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs({u=
   u(x,D),v=v(x,D)},subs(system_bi_derivate_binary_tree,dxyv)),
   x)))),
   dxyyu=factor(subs(system_derivate_binary_tree,subs({diff(u(x,
   D),x)=dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs({u=
   u(x,D),v=v(x,D)},subs(system_bi_derivate_binary_tree,dyyu)),
   x)))),
   dxyyv=factor(subs(system_derivate_binary_tree,subs({diff(u(x,
   D),x)=dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs({u=
   u(x,D),v=v(x,D)},subs(system_bi_derivate_binary_tree,dyyv)),
   x)))),
   dyyyu=factor(subs(system_derivate_binary_tree,subs({diff(u(x,
   D),D)=dyu,diff(v(x,D),D)=dyv,u(x,D)=u,v(x,D)=v},diff(subs({u=
   u(x,D),v=v(x,D)},subs(system_bi_derivate_binary_tree,dyyu)),
   D)))),
   dyyyv=factor(subs(system_derivate_binary_tree,subs({diff(u(x,
```

```
D),D)=dyu,diff(v(x,D),D)=dyv,u(x,D)=u,v(x,D)=v},diff(subs({u=
u(x,D),v=v(x,D)},subs(system_bi_derive_binary_tree,dyyv)),
D))))
}:
```

Expressions of generating functions of 3-connected networks

K is the generating function of networks such that the associated graph, obtained by adding the root edge, is 3-connected. K is equal to M/(2*x^2*y). We obtain expressions of K and its partial derivatives with respect to the generating functions of bicolored binary trees. As they will be evaluated at (x,D) in the systems given after, we replace y by D. These expressions will be placed in the systems verified by networks, pointed networks, and bi-pointed networks.

```
> system_3_connected:={K=subs(y=D,1/(2*x^2*y)*x^2*y^2*(1/(1+x*
y)+1/(1+y)-1-(1+u)^2*(1+v)^2/(1+u+v)^3))};
```

$$system\_3\_connected := \left\{K = \frac{\mathrm{D}\left(\frac{1}{\mathrm{D}\,x+1}+\frac{1}{1+\mathrm{D}}-1-\frac{(1+u)^2\,(1+v)^2}{(1+u+v)^3}\right)}{2}\right\} \quad (1.3.1)$$

```
> system_derivate_3_connected:={
dxK=subs(system_derivate_binary_tree,subs({diff(u(x,D),x)=
dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs(u=u(x,D),
v=v(x,D),subs(system_3_connected,K)),x))),
dyK=subs(system_derivate_binary_tree,subs({diff(u(x,D),D)=
dyu,diff(v(x,D),D)=dyv,u(x,D)=u,v(x,D)=v},diff(subs(u=u(x,D),
v=v(x,D),subs(system_3_connected,K)),D)))
}:
> system_bi_derivate_3_connected:={
dxxK=subs(system_derivate_binary_tree,subs({diff(u(x,D),x)=
dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs(u=u(x,D),
v=v(x,D),subs(system_derivate_3_connected,dxK)),x))),
dxyK=subs(system_derivate_binary_tree,subs({diff(u(x,D),x)=
dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs(u=u(x,D),
v=v(x,D),subs(system_derivate_3_connected,dyK)),x))),
dyyK=subs(system_derivate_binary_tree,subs({diff(u(x,D),D)=
dyu,diff(v(x,D),D)=dyv,u(x,D)=u,v(x,D)=v},diff(subs(u=u(x,D),
v=v(x,D),subs(system_derivate_3_connected,dyK)),D)))
}:
> system_tri_derivate_3_connected:={
dxxxK=subs(system_derivate_binary_tree,subs({diff(u(x,D),x)=
dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs(u=u(x,D),
v=v(x,D),subs(system_bi_derivate_3_connected,dxxK)),x))),
dxxyK=subs(system_derivate_binary_tree,subs({diff(u(x,D),x)=
dxu,diff(v(x,D),x)=dxv,u(x,D)=u,v(x,D)=v},diff(subs(u=u(x,D),
v=v(x,D),subs(system_bi_derivate_3_connected,dxyK)),x))),
dxyyK=subs(system_derivate_binary_tree,subs({diff(u(x,D),D)=
dyu,diff(v(x,D),D)=dyv,u(x,D)=u,v(x,D)=v},diff(subs(u=u(x,D),
v=v(x,D),subs(system_bi_derivate_3_connected,dxyK)),D))),
dyyyK=subs(system_derivate_binary_tree,subs({diff(u(x,D),D)=
dyu,diff(v(x,D),D)=dyv,u(x,D)=u,v(x,D)=v},diff(subs(u=u(x,D),
v=v(x,D),subs(system_bi_derivate_3_connected,dyyK)),D)))
}:
```

Evaluation of generating functions of networks

Networks

```
> system_network:={
  D=y+S+P+H,
  S=(y+P+H)*x*D,
  P=y*(exp(S+H)-1)+(exp(S+H)-S-H-1),
  H=K,

  u=x*D*(1+v)^2,
  v=D*(1+u)^2,
  K=1/2*D*(1/(1+x*D)+1/(1+D)-1-(1+u)^2*(1+v)^2/(1+u+v)^3)
  }:
> eval_network:=proc(x0,y0)
  global system_network:
  fsolve(subs({x=x0,y=y0},system_network),{u,v,K,S,P,H,D});
  end proc:
```

Derivate networks

```
> equation_derivate_network:={
  dD=dS+dP+dH,
  dS=(dP+dH)*x*D+(y+P+H)*D+(y+P+H)*x*dD,
  dP=y*(dS+dH)*exp(S+H)+(dS+dH)*(exp(S+H)-1),
  dH=dxK+dD*dyK
  }:
> system_derivate_network:=solve(equation_derivate_network,
  {dD,dS,dP,dH}):
> eval_derivate_network:=proc(x0,y0)
  local solutions_network, solutions_derivate_3_connected:
  global system_derivate_network,
  system_derivate_3_connected:

  solutions_network:=eval_network(x0,y0):
  solutions_derivate_3_connected:=evalf(subs({x=x0,y=y0,op
  (solutions_network)},system_derivate_3_connected)):
  evalf(subs({x=x0,y=y0,op(solutions_network),op
  (solutions_derivate_3_connected)},system_derivate_network)
  ):

  end proc:
```

Bi-derivate networks

```
> equation_bi_derivate_network:={
  ddD=ddS+ddP+ddH,
  ddS=(ddP+ddH)*x*D+2*(dP+dH)*D+2*(dP+dH)*x*dD+2*(y+P+H)*dD+
  (y+P+H)*x*ddD,
  ddP=y*(ddS+ddH)*exp(S+H)+y*(dS+dH)^2*exp(S+H)+(ddS+ddH)*
  (exp(S+H)-1)+(dS+dH)^2*exp(S+H),
  ddH=dxxK+dD*dxyK+ddD*dyK+dD*dxyK+dD^2*dyyK
  }:

> system_bi_derivate_network:=solve
  (equation_bi_derivate_network,{ddD,ddS,ddP,ddH}):
> eval_bi_derivate_network:=proc(x0,y0)

  local solutions_network, solutions_derivate_network,
```

```
  solutions_derivate_3_connected,
  solutions_bi_derivate_3_connected:

  global system_bi_pointed_network,
  system_derivate_3_connected,
  system_bi_derivate_3_connected:

  solutions_network:=eval_network(x0,y0):

  solutions_derivate_3_connected:=evalf(subs({x=x0,y=y0,op
  (solutions_network)},system_derivate_3_connected)):
  solutions_bi_derivate_3_connected:=evalf(subs({x=x0,y=y0,
  op(solutions_network)},system_bi_derivate_3_connected)):

  solutions_derivate_network:=evalf(subs({x=x0,y=y0,op
  (solutions_network),op(solutions_derivate_3_connected)},
  system_derivate_network)):

  evalf(subs({x=x0,y=y0,op(solutions_network),op
  (solutions_derivate_network),op
  (solutions_derivate_3_connected),op
  (solutions_bi_derivate_3_connected)},
  system_bi_derivate_network)):
  end proc:
```

Tri-derivate networks

```
> equation_bi_derivate_network:={
  ddD=ddS+ddP+ddH,
  ddS=(ddP+ddH)*x*D+2*(dP+dH)*D+2*(dP+dH)*x*dD+2*(y+P+H)*dD+
  (y+P+H)*x*ddD,
  ddP=y*(ddS+ddH)*exp(S+H)+y*(dS+dH)^2*exp(S+H)+(ddS+ddH)*
  (exp(S+H)-1)+(dS+dH)^2*exp(S+H),
  ddH=dxxK+dD*dxyK+ddD*dyK+dD*dxyK+dD^2*dyyK
  }:

> equation_tri_derivate_network:={
  dddD=dddS+dddP+dddH,
  dddS=(dddP+dddH)*x*D+(ddP+ddH)*D+(ddP+ddH)*x*dD +2*(ddP+
  ddH)*D+2*(dP+dH)*dD   + 2*(ddP+ddH)*x*dD+2*(dP+dH)*dD+2*
  (dP+dH)*x*ddD+2*(dP+dH)*dD+2*(y+P+H)*ddD+(dP+dH)*x*ddD+(y+
  P+H)*ddD+(y+P+H)*x*dddD,
  dddP=y*(dddS+dddH)*exp(S+H)+y*(ddS+ddH)*(dS+dH)*exp(S+H)
  +2*y*(ddS+ddH)*(dS+dH)*exp(S+H)+y*(dS+dH)^3*exp(S+H)   +
  (dddS+dddH)*(exp(S+H)-1)+(ddS+ddH)*(dS+dH)*exp(S+H)   +2*
  (ddS+ddH)*(dS+dH)*exp(S+H)+(dS+dH)^3*exp(S+H),
  dddH=
  dxxxK+dD*dxxyK+ddD*dxyK+dD*dxxyK+dD^2*dxyyK +dddD*dyK+ddD*
  dxyK+ddD*dD*dyyK   +ddD*dxyK+dD*dxxyK+dD^2*dxyyK   +2*ddD*
  dD*dyyK+dD^2*dxyyK+dD^3*dyyyK
  }:

> system_tri_derivate_network:=solve
  (equation_tri_derivate_network,{dddD,dddS,dddP,dddH}):
> eval_tri_derivate_network:=proc(x0,y0)

  local solutions_network, solutions_derivate_network,
```

```
solutions_bi_derivate_network,
solutions_derivate_3_connected,
solutions_bi_derivate_3_connected,
solutions_tri_derivate_3_connected,
system_tri_derivate_networks:

global system_bi_pointed_network,
system_tri_pointed_network, system_derivate_3_connected,
system_bi_derivate_3_connected:

solutions_network:=eval_network(x0,y0):
solutions_derivate_3_connected:=evalf(subs({x=x0,y=y0,op
(solutions_network)},system_derivate_3_connected)):
solutions_bi_derivate_3_connected:=evalf(subs({x=x0,y=y0,
op(solutions_network)},system_bi_derivate_3_connected)):
solutions_tri_derivate_3_connected:=evalf(subs({x=x0,y=y0,
op(solutions_network)},system_tri_derivate_3_connected)):
solutions_derivate_network:=evalf(subs({x=x0,y=y0,op
(solutions_network),op(solutions_derivate_3_connected)},
system_derivate_network)):
solutions_bi_derivate_network:=evalf(subs({x=x0,y=y0,op
(solutions_network),op(solutions_derivate_network),op
(solutions_derivate_3_connected),op
(solutions_bi_derivate_3_connected)},
system_bi_derivate_network)):
system_tri_derivate_networks:=subs({x=x0,y=y0,op
(solutions_network),op(solutions_derivate_network),op
(solutions_bi_derivate_network),op
(solutions_derivate_3_connected),op
(solutions_bi_derivate_3_connected),op
(solutions_tri_derivate_3_connected)},
equation_tri_derivate_network):
fsolve(system_tri_derivate_networks,{dddD,dddS,dddP,dddH})
:
end proc:
```

Bender-Gao-Wormald singularity analysis of x->D(x,y)

Finding the singularity of x->D(x,y)

We follow the notations of Gimenez-Noy for the system giving the singularity of x->D(x,y)

```
> system_singularity_networks:={
  xsi=(1+3*t)*(1-t)^3/(16*t^3),
  Y=(1+2*t)/((1+3*t)*(1-t))*exp(-(t^2*(1-t)*(18+36*t+5*t^2))
  /(2*(3+t)*(1+2*t)*(1+3*t)^2))-1
  }:
```

```
> find_singularity:=proc(Y0)
  fsolve(subs(Y=Y0,system_singularity_networks),{t,xsi});
  end proc:
```

```
> find_singularity(1);
```

$$\{t = 0.62637166330645166589299785045039561167 21, xsi = 0.038191097669411335391152564045422359553 88\} \quad \textbf{(1.5.1.1)}$$

Development of x->D(x,y) near the singularity

```
> system_alpha_beta:={
  alpha=144+592*t+664*t^2+135*t^3+6*t^4-5*t^5,
  beta=3*t*(1+t)*(400+1808*t+2527*t^2+1155*t^3+237*t^4+17*
  t^5)
  }:

  system_D0_D2_D3:={
  D_0=3*t^2/((1-t)*(1+3*t)),
  D_2=-(48*t^2*(1+t)*(1+2*t)^2*(18+6*t+t^2))/((1+3*t)*beta),
  D_3=384*t^3*(1+t)^2*(1+2*t)^2*(3+t)^2*alpha^(3/2)*beta^
  (-5/2)
  }:
> find_D0_D1_D2:=proc(Y0)
  global system_singularity_networks, system_alpha_beta:
  local solutions_singularity:
  solutions_singularity:=fsolve(subs(Y=Y0,
  system_singularity_networks),{t,xsi}):
  evalf(fsolve(subs(solutions_singularity,subs
  (system_alpha_beta,system_D0_D2_D3)),{D_0,D_2,D_3}));
  end proc:
> find_D0_D1_D2(1);
```

$$\{D\_0 = 1.094174633098579501442769060881886356836,\ D\_2 = -0.13749382418806273997533867902060211970 97,\ D\_3 = 0.054329404911861934541444154009027221475 75\}$$ **(1.5.2.1)**

Comparison between the development of x->D(x,y) and the evaluation-procedures near the singularity

Here we check that the approximate evluation of x->D(x,y) given by the development of Bender-Gao-Wormald corresponds to the exact evaluation given by our procedures. The argument k stands for the decimal where we trunk the evaluation of the singularity (indeed we have to evaluate at a value near below the singularity).

Networks

```
> compare_network_exact_evaluation_approximate_evaluation:=
  proc(Y0,k)
  local Rk, solutions_singularity, compare:
  solutions_singularity:=find_singularity(Y0):
  Rk:=subs(solutions_singularity,xsi)-10^(-k):
  compare[1]:=subs(fsolve(subs(solutions_singularity,subs
  (system_alpha_beta,system_D0_D2_D3)),{D_0,D_2,D_3}),D_0)
  :
  compare[2]:=subs(eval_network(Rk,Y0),D):
  approx=compare[1],exact=compare[2];
  end proc:
> compare_network_exact_evaluation_approximate_evaluation
  (2,18);
```

$$approx = 2.102839095761082523275532501236260941661,\ exact = 2.102839095761082503744830134737258811570$$ **(1.5.3.1.1)**

Pointed networks

```
> compare_derivate_network_exact_evaluation_approximate_eva
  luation:=proc(Y0,k)
  local Rk, R, D2, solutions_singularity,
  solutions_D0_D2_D3, compare:
  solutions_singularity:=find_singularity(Y0):
```

```
  R:=subs(solutions_singularity,xsi):
  Rk:=R-10^(-k):
  solutions_D0_D2_D3:=fsolve(subs(solutions_singularity,
  subs(system_alpha_beta,system_D0_D2_D3)),{D_0,D_2,D_3}):
  D2:=subs(solutions_D0_D2_D3,D_2):
  compare[1]:=-D2/R:
  compare[2]:=subs(eval_derate_network(Rk,Y0),dD):
  approx=compare[1],exact=compare[2];
  end proc:
> compare_derivate_network_exact_evaluation_approximate_eva
  luation(2,12);
```

$approx = 19.530702453012013421256695119929254558 72$, $exact$ **(1.5.3.2.1)**
$= 19.530572684475986805947661504909095451 97$

Bi-pointed networks

```
> compare_bi_derivate_network_exact_evaluation_approximate_
  evaluation:=proc(Y0,k)
  local Rk, R, D3, solutions_singularity,
  solutions_D0_D2_D3, compare:
  solutions_singularity:=find_singularity(Y0):
  R:=subs(solutions_singularity,xsi):
  Rk:=R-10^(-k):
  solutions_D0_D2_D3:=fsolve(subs(solutions_singularity,
  subs(system_alpha_beta,system_D0_D2_D3)),{D_0,D_2,D_3}):
  D3:=subs(solutions_D0_D2_D3,D_3):
  compare[1]:=3*D3/(4*R^2)*(1-Rk/R)^(-1/2):
  compare[2]:=subs(eval_bi_derivate_network(Rk,Y0),ddD):
  approx=compare[1],exact=compare[2];
  end proc:
> compare_bi_derivate_network_exact_evaluation_approximate_
  evaluation(2,15);
```

$approx = 2.05183623354063790520396401243481280637 0\ 10^{9}$, $exact$ **(1.5.3.3.1)**
$= 2.05183525188255243166847569596386692790 5\ 10^{9}$

Tri-pointed networks

```
> compare_tri_derivate_network_exact_evaluation_approximate
  _evaluation:=proc(Y0,k)
  local Rk, R, D3, solutions_singularity,
  solutions_D0_D2_D3, compare:
  solutions_singularity:=find_singularity(Y0):
  R:=subs(solutions_singularity,xsi):
  Rk:=R-10^(-k):
  solutions_D0_D2_D3:=fsolve(subs(solutions_singularity,
  subs(system_alpha_beta,system_D0_D2_D3)),{D_0,D_2,D_3}):
  D3:=subs(solutions_D0_D2_D3,D_3):
  compare[1]:=3*D3/(8*R^3)*(1-Rk/R)^(-3/2):
  compare[2]:=subs(eval_tri_derivate_network(Rk,Y0),dddD):
  approx=compare[1],exact=compare[2];
  end proc:


> compare_tri_derivate_network_exact_evaluation_approximate
  _evaluation(2,15);
```

$approx = 1.02591811677031895260198200602120176179 5\ 10^{24}$, $exact$ **(1.5.3.4.1)**
$= 1.02591811677002741708815590845619862415 3\ 10^{24}$

When generating only networks, calculation of the choose-vectors for the Boltzmann samplers of binary trees and networks

Choose of the size of the networks

The real y0 has to be chosen to give a desired balance between the number of vertices and number of edges. See the curve of mu(t) given by Bender, Gao and Wormald

```
> y0:=1;
```

$$y0 := 1 \qquad (1.6.1.1)$$

Then we choose a size N around which we want the boltzmann sampler for bi-pointed networks to have a good chance of picking up networks of this size

```
> N:=1000000:
```

We calculate the real x0 for which the Boltzmann sampler of bi-pointed networks has good chances of picking up networks of this size. This value x0 verifies x=R(1-1/2N) where R is the singularity of x->D(x,y0)

```
> R:=subs(find_singularity(y0),xsi); x0:=evalf(R*(1-1/(2*N)))
  );
```

$$R := 0.038191097669411335391152564045422359553 88$$
$$x0 := 0.038191078573862500685484868469140336842 70 \qquad (1.6.1.2)$$

Calculating the choose-vectors of bicolored binary trees

```
> solutions_networks:=eval_network(x0,y0);
```

$$solutions\_networks := \{D = 1.094174564370870916793206753393405522040,\ H = 0.002123268318091556372016192229859650620518,\ K = 0.002123268318091556372016192229859650620518,\ P = 0.048162271883103056991289466417157242774 90,\ S = 0.043889024169676303429901094746388628645 03,\ u = 0.531504947402927938746237404454401716907 5,\ v = 2.566394541912430019048846715169368291351\} \qquad (1.6.2.1)$$

```
> u_eval:=subs(solutions_networks,u):v_eval:=subs
  (solutions_networks,v):
> ch_1_or_u:=[evalf(1/(1+u_eval))];
```

$$ch\_1\_or\_u := [0.652952510336819167265023650563038 6828934] \qquad (1.6.2.2)$$

```
> ch_1_or_v:=[evalf(1/(1+v_eval))];
```

$$ch\_1\_or\_v := [0.280395224994866595584755918792205 4010592] \qquad (1.6.2.3)$$

```
> ch_u_or_v:=[evalf(u_eval/(u_eval+v_eval))];
```

$$ch\_u\_or\_v := [0.171569461577461186547607546703498 7317219] \qquad (1.6.2.4)$$

Calculating the choose-vectors of pointed bicolored binary trees

```
> solution_derivate_binary_tree:=subs({x=x0,op
  (solutions_networks)},system_derivate_binary_tree);
```

$$solution\_derivate\_binary\_tree := \{dxu = 13230.625331547239353911536224560914356 13,\ dxv = 44342.011032417321120430334811027485858 96,\ dyu = 1126.431512981459152507965343199629438 780,\ dyv = 3777.544133638864258208363721570382908 734\} \qquad (1.6.3.1)$$

Here pU and pV stand for the generating functions of black-rooted and white-rooted bicolored binary trees with a pointed black vertex

```
> dxu_eval:=evalf(subs(solution_derivate_binary_tree,dxu))
  :dxv_eval:=evalf(subs(solution_derivate_binary_tree,dxv))
```

```
:dyu_eval:=evalf(subs(solution_derivate_binary_tree,dyu))
:dyv_eval:=evalf(subs(solution_derivate_binary_tree,dyv)):
> ch_dxu_or_dxv:=[evalf(dxu_eval/(dxu_eval+dxv_eval))];
```

$$ch\_dxu\_or\_dxv := [0.22980752953374312882718067053332124431 82]$$ **(1.6.3.2)**

```
> D_eval:=subs(solutions_networks,D);
```

$$D\_eval := 1.094174564370870916793206753393405522040$$ **(1.6.3.3)**

```
> choose_vector_dxu_Bernoulli:=[(1+v_eval)^2*
  D_eval/dxu_eval,x0*D_eval*dxv_eval*(1+v_eval)/dxu_eval,(1+
  v_eval)*x0*D_eval*dxv_eval/dxu_eval];
  choose_vector_dxu:=[choose_vector_dxu_Bernoulli[1],
  choose_vector_dxu_Bernoulli[1]+choose_vector_dxu_Bernoulli
  [2]];
```

$$choose\_vector\_dxu\_Bernoulli := [0.00105187713932158383879344925179959133670 0,\ 0.49947406143033920808060327537410020433 17,\ 0.49947406143033920808060327537410020433 17]$$

$$choose\_vector\_dxu := [0.00105187713932158383879344925179959133670 0,\ 0.50052593856966079191939672462589979566 84]$$ **(1.6.3.4)**

```
> choose_vector_dxv:=[0.5];
```

$$choose\_vector\_dxv := [0.5]$$ **(1.6.3.5)**

```
> ch_dyu_or_dyv:=[evalf(dyu_eval/(dyu_eval+dyv_eval))];
```

$$ch\_dyu\_or\_dyv := [0.22969761559843038913952606839667943602 21]$$ **(1.6.3.6)**

```
> choose_vector_dyv_Bernoulli:=[(1+u_eval)^2/dyv_eval,
  dyu_eval*D_eval*(1+u_eval)/dyv_eval,(1+u_eval)*D_eval*
  dyu_eval/dyv_eval];
  choose_vector_dyv:=[choose_vector_dyv_Bernoulli[1],
  choose_vector_dyv_Bernoulli[1]+choose_vector_dyv_Bernoulli
  [2]];
```

$$choose\_vector\_dyv\_Bernoulli := [0.00062090800820379697680009889039482160311 75,\ 0.49968954599589810151159995055480258919 83,\ 0.49968954599589810151159995055480258919 83]$$

$$choose\_vector\_dyv := [0.00062090800820379697680009889039482160311 75,\ 0.50031045400410189848840004944519741080 14]$$ **(1.6.3.7)**

```
> choose_vector_dyu_Bernoulli:=[x0*(1+v_eval)^2/dyu_eval,x0*
  D_eval*dyv_eval*(1+v_eval)/dyu_eval,x0*D_eval*(1+v_eval)*
  dyv_eval/dyu_eval];
  choose_vector_dyu:=[choose_vector_dyu_Bernoulli[1],
  choose_vector_dyu_Bernoulli[1]+choose_vector_dyu_Bernoulli
  [2]];
```

$$choose\_vector\_dyu\_Bernoulli := [0.00043123688955594504460993276563503883165 94,\ 0.49978438155522202747769503361718248058 44,\ 0.49978438155522202747769503361718248058 44]$$

$$choose\_vector\_dyu := [0.00043123688955594504460993276563503883165 94,\ 0.50021561844477797252230496638281751941 61]$$ **(1.6.3.8)**

```
> ch_b_or_dxb:=[evalf((u_eval+v_eval)/(u_eval+v_eval+x0*
  dxu_eval+x0*dxv_eval))];
```

$$ch\_b\_or\_dxb := [0.00140694727476481448795333148854065549 3055]$$ **(1.6.3.9)**

```
> ch_3b_or_dyb:=[evalf((3*u_eval+3*v_eval)/(3*u_eval+3*
  v_eval+D_eval*dyu_eval+D_eval*dyv_eval))];
```

$$ch\_3b\_or\_dyb := [0.00172902829670659371928471391190777038555 8]$$ **(1.6.3.10)**

Calculating the choose-vectors of 3-connected networks

```
> solution_derivate_3_connected:=subs({x=x0,op(eval_network
  (x0,y0))},system_derivate_3_connected);
  solution_bi_derivate_3_connected:=subs({x=x0,op
  (eval_network(x0,y0))},system_bi_derivate_3_connected);
```

$$solution\_derivate\_3\_connected := \{dxK = 0.25392572459989240542084343932509638427 48, dyK = 0.02180756983520382597105840098439961183860\}$$

$$solution\_bi\_derivate\_3\_connected := \{dxxK = 8711.47166229600977258508799176668118161 0, dxyK = 742.35971493554108180946743424694847979 65, dyyK = 63.19442031482353372038518425764293817000\} \quad (1.6.4.1)$$

```
> K_eval:=subs(solutions_networks,K);dxK_eval:=subs
  (solution_derivate_3_connected,dxK);dyK_eval:=subs
  (solution_derivate_3_connected,dyK);dxyK_eval:=subs
  (solution_bi_derivate_3_connected,dxyK);
```

$$K\_eval := 0.00212326831809155637201619222985965062051 8$$

$$dxK\_eval := 0.25392572459989240542084343932509638427 48$$

$$dyK\_eval := 0.02180756983520382597105840098439961183860$$

$$dxyK\_eval := 742.35971493554108180946743424694847979 65 \quad (1.6.4.2)$$

```
> ch_K_in_dyK:=[evalf(K_eval/(y0*dyK_eval))];
```

$$ch\_K\_in\_dyK := [0.09736382064286582607785119655750788825036] \quad (1.6.4.3)$$

```
> ch_dxK_in_dxyK:=[evalf(dxK_eval/(y0*dxyK_eval))];
```

$$ch\_dxK\_in\_dxyK := [0.00034205213387951785063590145517007616783 39] \quad (1.6.4.4)$$

Calculating the choose-vectors of networks

We recall the system verified by the networks

```
> system_verified_by_network:={
  D=y+S+P+H,
  S=(y+P+H)*x*D,
  P=y*(exp(S+H)-1)+(exp(S+H)-S-H-1),
  H=K}:
> solutions_networks:=eval_network(x0,y0);
```

$$solutions\_networks := \{D = 1.09417456437087091679320675339340552204 0, H = 0.00212326831809155637201619222985965062051 8, K = 0.00212326831809155637201619222985965062051 8, P = 0.04816227188310305699128946641715724277490, S = 0.04388902416967630342990109474638862864503, u = 0.53150494740292793874623740445440171690 75, v = 2.56639454191243001904884671516936829135 1\} \quad (1.6.5.1)$$

```
> Deval:=subs(solutions_networks,D):Seval:=subs
  (solutions_networks,S):Peval:=subs(solutions_networks,P)
  :Heval:=subs(solutions_networks,H):
> choose_vector_non_trivial_D_Bernoulli:=[evalf(Seval/
  (Deval-y0)),evalf(Peval/(Deval-y0)),evalf(Heval/(Deval-y0)
  )];
  choose_vector_non_trivial_D:=
  [choose_vector_non_trivial_D_Bernoulli[1],
  choose_vector_non_trivial_D_Bernoulli[1]+
  choose_vector_non_trivial_D_Bernoulli[2]];
```

$$choose\_vector\_non\_trivial\_D\_Bernoulli := [0.46603904634839589009660545863589332516 95,$$

0.5114148624402886335251618783591683374389,
0.022546091211315476378232663004938337396311]

choose_vector_non_trivial_D := **(1.6.5.2)**

[0.46603904634839589009660545863589332516951,
0.977453908788684523621767336995060166260841]

```
> choose_vector_D_Bernoulli:=[evalf(y0/Deval),evalf
  (Seval/Deval),evalf(Peval/Deval),evalf(Heval/Deval)];
  choose_vector_D[1]+choose_vector_D[2]+choose_vector_D[3]+
  choose_vector_D[4];
  choose_vector_D:=[choose_vector_D_Bernoulli[1],
  choose_vector_D_Bernoulli[1]+choose_vector_D_Bernoulli[2],
  choose_vector_D_Bernoulli[1]+choose_vector_D_Bernoulli[2]+
  choose_vector_D_Bernoulli[3]];
```

choose_vector_D_Bernoulli :=

[0.91393095083962268965377078684537679053531,
0.040111537590815445705800737938285485272981,
0.044016990936720802082280492528049579076821,
0.001940520632841062558147982688288145115344]

$choose\_vector\_D_1 + choose\_vector\_D_2 + choose\_vector\_D_3 + choose\_vector\_D_4$

choose_vector_D := [0.91393095083962268965377078684537679053531, **(1.6.5.3)**
0.954042488430438135359571524783662275808 3,
0.998059479367158937441852017311711854885 1]

```
> choose_vector_P:=[evalf(y0*(exp(Seval+Heval)-1))/Peval];
```

choose_vector_P := [0.97767983827098610986898981012310840309 50] **(1.6.5.4)**

```
> ch_y_or_P_or_H_Bernoulli:=[evalf(y0/(y0+Peval+Heval)),
  evalf(Peval/(y0+Peval+Heval)),evalf(Heval/(y0+Peval+Heval)
  )];
  ch_y_or_P_or_H:=[ch_y_or_P_or_H_Bernoulli[1],
  ch_y_or_P_or_H_Bernoulli[1]+ch_y_or_P_or_H_Bernoulli[2]];
```

ch_y_or_P_or_H_Bernoulli :=

[0.95212202941348519033925565531451712737 71,
0.045856360046504119595405010510193338202 09,
0.002021610540010690065339334175289534420293]

ch_y_or_P_or_H := [0.95212202941348519033925565531451712737 71, **(1.6.5.5)**
0.997978389459989309934660665824710465579 2]

```
> ch_S_or_H:=[evalf(Seval/(Seval+Heval))];
```

ch_S_or_H := [0.95385432450130545561815998820163951601 76] **(1.6.5.6)**

We compute the vector for the Poisson laws of parameter S+H. Notice that the 1+... is only used to have an output without exponent notation. When put in the file, the first 1 has to be replaced by a 0.

```
> exp_S_plus_H:=evalf(exp(Seval+Heval)):
  poisson_S_plus_H:=[seq(0,i=1..21)]:
  poisson_S_plus_H[1]:=evalf(1/exp_S_plus_H):
  for i from 2 to 21 do poisson_S_plus_H[i]:=
  poisson_S_plus_H[i-1]+evalf((Seval+Heval)^(i-1)/(i-1)!
  /exp_S_plus_H);
  od:
  poisson_S_plus_H;
```

[0.95503022242123224730630308451852882409 03, **(1.6.5.7)**
0.998973352349935980105730409838607548316 3,
0.999984314423489730992419722564588585567 6,
0.999999819984363862815903665606290938704 2,

```
    0.999999998345964394676641475423760922335 3,
    0.999999999873296211283190221659581348341,
    0.999999999999916783941443341304319751522 3,
    0.999999999999999521686721717171581381923 6,
    0.999999999999999997555888658054660065043 9,
    0.999999999999999999887588019651427382331,
    0.999999999999999999999529952297794396222,
    0.999999999999999999999998198202028091558,
    0.999999999999999999999999993624319908235,
    0.999999999999999999999999999979050341322,
    0.999999999999999999999999999999935749563,
    0.999999999999999999999999999999999815264,
    0.999999999999999999999999999999999999502,
    1.000000000000000000000000000000000000000,
    1.000000000000000000000000000000000000000,
    1.000000000000000000000000000000000000000,
    1.000000000000000000000000000000000000000]
> exp_at_least_1_S_plus_H:=evalf(exp(Seval+Heval)-1):
  poisson_at_least_1_S_plus_H:=[seq(0,i=1..21)]:
  poisson_at_least_1_S_plus_H[1]:=evalf((Seval+Heval)
  /exp_at_least_1_S_plus_H):
  for i from 2 to 21 do
  poisson_at_least_1_S_plus_H[i]:=
  poisson_at_least_1_S_plus_H[i-1]+evalf((Seval+Heval)^i/i!
  /exp_at_least_1_S_plus_H);
  od:
  poisson_at_least_1_S_plus_H;
[0.977170275119422732949676243281663325131 7,                    (1.6.5.8)
    0.999651197373996466658161271439529760441 1,
    0.999995996964053872093298531181184285816 5,
    0.999999963218950718041735231617012804390 8,
    0.999999999718246796985199421582304403948 7,
    0.999999999998149511448863631332802981464 7,
    0.999999999999989363672581101189328386890 3,
    0.999999999999999945649912595980585151097 3,
    0.999999999999999999750027715499203733419 5,
    0.999999999999999999998954747549324027872 6,
    0.999999999999999999999959933135787641720,
    0.999999999999999999999999858223001423642,
    0.999999999999999999999999995341391528866,
    0.999999999999999999999999999998571252956,
    0.999999999999999999999999999999995891998,
    0.999999999999999999999999999999999988930,
    1.000000000000000000000000000000000000002,
    1.000000000000000000000000000000000000005,
    1.000000000000000000000000000000000000005,
    1.000000000000000000000000000000000000005,
    1.000000000000000000000000000000000000005]
> exp_at_least_2_S_plus_H:=evalf(exp(Seval+Heval)-Seval-
  Heval-1):
  poisson_at_least_2_S_plus_H:=[seq(0,i=1..21)]:
  poisson_at_least_2_S_plus_H[1]:=evalf((Seval+Heval)
  ^2/2/exp_at_least_2_S_plus_H):
```

```
for i from 2 to 21 do
poisson_at_least_2_S_plus_H[i]:=
poisson_at_least_2_S_plus_H[i-1]+evalf((Seval+Heval)^(i+1)
/(i+1)!/exp_at_least_2_S_plus_H);
od:
poisson_at_least_2_S_plus_H;
```

$$[0.98472155806396587637157844913103405285 25,\ 0.99982465684684910285801679211064320294 49,\ 0.99999838889651652130519179918773821389 19,\ 0.99999998765849328063929637520627804168 67,\ 0.99999999991894389613469675552183181340 76,\ 0.99999999999953410181355501893372913838 26,\ 0.99999999999999761932797314353219090735 69,\ 0.99999999999999998905057832241053593387 23,\ 0.99999999999999999954215284847114543488 85,\ 0.99999999999999999999824496946757139096 8,\ 0.99999999999999999999999378980695922095 0,\ 0.99999999999999999999999997959411033404 9,\ 0.99999999999999999999999999937417250766,\ 0.99999999999999999999999999999820063301,\ 0.99999999999999999999999999999999519330,\ 1.00000000000000000000000000000000000505,\ 1.00000000000000000000000000000000000629,\ 1.00000000000000000000000000000000000629,\ 1.00000000000000000000000000000000000629,\ 1.00000000000000000000000000000000000629,\ 1.00000000000000000000000000000000000629]$$ **(1.6.5.9)**

Calculating the choose-vectors of derivate networks

We recall the system verified by pointed networks

```
> equations_derivate_network:={
dD=dS+dP+dH,
dS=(dP+dH)*x*D+(y+P+H)*D+(y+P+H)*x*dD,
dP=y*(dS+dH)*exp(S+H)+(dS+dH)*(exp(S+H)-1),
dH=dxK+dD*dyK
}:
```

```
> solution_derivate_network:=eval_derivate_network(x0,y0);
```

$$solution\_derivate\_network := \{dD = 3.59864541423366329023834517989935262332 8,\ dH = 0.33240343578292901802664306370630343525 95,\ dP = 1.88023784904830183671547258423502766503 8,\ dS = 1.38600412940243243549622953195802152303 1\}$$ **(1.6.6.1)**

```
> dDeval:=subs(solution_derivate_network,dD):dSeval:=subs
(solution_derivate_network,dS):dPeval:=subs
(solution_derivate_network,dP):dHeval:=subs
(solution_derivate_network,dH):dxKeval:=subs
(solution_derivate_3_connected,dxK):dyKeval:=subs
(solution_derivate_3_connected,dyK):
```

```
> choose_vector_dD_Bernoulli:=[evalf(dSeval/dDeval),evalf
(dPeval/dDeval),evalf(dHeval/dDeval)];
choose_vector_dD:=[choose_vector_dD_Bernoulli[1],
choose_vector_dD_Bernoulli[1]+choose_vector_dD_Bernoulli
[2]];
```

choose_vector_dD_Bernoulli :=
[0.385146067439818603410722680150780500719 7,
0.522484888789383876346848457740735587955 1,
0.092369043770797520242428862108483911325 36]
choose_vector_dD := [0.385146067439818603410722680150780500719 7, 0.907630956229202479757571137891516088674 8] **(1.6.6.2)**

```
> choose_vector_dS_Bernoulli:=[evalf((dPeval+dHeval)*x0*
  Deval/dSeval),evalf((y0+Peval+Heval)*Deval/dSeval),evalf(
  (y0+Peval+Heval)*x0*dDeval/dSeval)];
  choose_vector_dS:=[choose_vector_dS_Bernoulli[1],
  choose_vector_dS_Bernoulli[1]+choose_vector_dS_Bernoulli
  [2]];
```

choose_vector_dS_Bernoulli :=
[0.066710627491838994078973066000735697501 44,
0.829143073267852352260199138386321945810 2,
0.104146299240308653660827795612942356689 3]
choose_vector_dS := [0.066710627491838994078973066000735697501 44, 0.895853700759691346339172204387057643311 6] **(1.6.6.3)**

```
> choose_vector_dP:=[evalf(y0*(dSeval+dHeval)*exp(Seval+
  Heval)/dPeval)];
```

choose_vector_dP := [0.956965475419811344826885393422688395266 6] **(1.6.6.4)**

```
> choose_vector_dH:=[evalf(dxKeval/dHeval)];
```

choose_vector_dH := [0.763908243011407131991473658184345425014 5] **(1.6.6.5)**

```
> ch_dP_or_dH:=[evalf(dPeval/(dPeval+dHeval))];
```

ch_dP_or_dH := [0.849770752239993985394774817190968580721 9] **(1.6.6.6)**

```
> ch_dS_or_dH:=[evalf(dSeval/(dSeval+dHeval))];
```

ch_dS_or_dH := [0.806563097999936195167442843686623781007 5] **(1.6.6.7)**

Calculating the choose-vectors of bi-derivate networks

We recall the system verified by bi-pointed networks

```
> equations_bi_derive_network:={
  ddD=ddS+ddP+ddH,
  ddS=(ddP+ddH)*x*D+2*(dP+dH)*D+2*(dP+dH)*x*dD+2*(y+P+H)*dD+
  (y+P+H)*x*ddD,
  ddP=y*(ddS+ddH)*exp(S+H)+y*(dS+dH)^2*exp(S+H)+(ddS+ddH)*
  (exp(S+H)-1)+(dS+dH)^2*exp(S+H),
  ddH=dxxK+2*dD*dxyK+ddD*dyK+dD^2*dyyK
  }:
> solution_bi_derive_network:=eval_bi_derive_network(x0,
  y0);solutions_bi_derive_3_connected:=subs({x=x0,op
  (eval_network(x0,y0))},system_bi_derive_3_connected);
```

solution_bi_derive_network := {*ddD*
= 39481.662538115071655969411352587448086 33, *ddH*
= 15733.833009982466248744792206267999180 63, *ddP*
= 20631.524985007123046509599739381136236 94, *ddS*
= 3116.304543125482360715019406938312668771}
solutions_bi_derive_3_connected := {*dxxK* **(1.6.7.1)**
= 8711.471662296009772585087991766681181610, *dxyK*
= 742.359714935541081809467434246948479796 5, *dyyK*
= 63.194420314823533720385184257642938170 00}

```
> ddDeval:=subs(solution_bi_derive_network,ddD):ddSeval:=
  subs(solution_bi_derive_network,ddS):ddPeval:=subs
```

```
(solution_bi_derive_network,ddP):ddHeval:=subs
(solution_bi_derive_network,ddH):dxxKeval:=subs
(solutions_bi_derive_3_connected,dxxK):dxyKeval:=subs
(solutions_bi_derive_3_connected,dxyK):dyyKeval:=subs
(solutions_bi_derive_3_connected,dyyK):
```

```
> choose_vector_ddD_Bernoulli:=[evalf(ddSeval/ddDeval),evalf
(ddPeval/ddDeval),evalf(ddHeval/ddDeval)];
choose_vector_ddD:=[choose_vector_ddD_Bernoulli[1],
choose_vector_ddD_Bernoulli[1]+choose_vector_ddD_Bernoulli
[2]];
```

$$choose\_vector\_ddD\_Bernoulli := [0.078930428527852478895475841654043401500 41, 0.522559681094729062056682071303913572354 3, 0.398509890377418459047842087042043026145 5]$$

$$choose\_vector\_ddD := [0.078930428527852478895475841654043401500 41, 0.601490109622581540952157912957956973854 7] \quad \textbf{(1.6.7.2)}$$

```
> choose_vector_ddS_Bernoulli:=[evalf((ddPeval+ddHeval)*x0*
Deval/ddSeval),evalf(2*(dPeval+dHeval)*Deval/ddSeval),
evalf(2*(dPeval+dHeval)*x0*dDeval/ddSeval),evalf(2*(y0+
Peval+Heval)*dDeval/ddSeval),evalf((y0+Peval+Heval)*x0*
ddDeval/ddSeval)];
choose_vector_ddS:=[choose_vector_ddS_Bernoulli[1],
choose_vector_ddS_Bernoulli[1]+choose_vector_ddS_Bernoulli
[2],choose_vector_ddS_Bernoulli[1]+
choose_vector_ddS_Bernoulli[2]+choose_vector_ddS_Bernoulli
[3],choose_vector_ddS_Bernoulli[1]+
choose_vector_ddS_Bernoulli[2]+choose_vector_ddS_Bernoulli
[3]+choose_vector_ddS_Bernoulli[4]];
```

$$choose\_vector\_ddS\_Bernoulli := [0.487636845224443478866747971533332483744 8, 0.001553773567657203488065862832599246908248, 0.000195165071199519814006349846205638307917 1, 0.002425696969327791219453529695394384938588, 0.508188519167372006611726286092468246101 0]$$

$$choose\_vector\_ddS := [0.487636845224443478866747971533332483744 8, 0.489190618792100682354813834365931730653 0, 0.489385783863300202168820184212137368960 9, 0.491811480832627993388273713907531753899 5] \quad \textbf{(1.6.7.3)}$$

```
> choose_vector_ddP_Bernoulli:=[evalf(y0*(ddSeval+ddHeval)*
exp(Seval+Heval)/ddPeval),evalf(y0*(dSeval+dHeval)^2*exp
(Seval+Heval)/ddPeval),evalf((ddSeval+ddHeval)*(exp(Seval+
Heval)-1)/ddPeval),evalf((dSeval+dHeval)^2*exp(Seval+
Heval)/ddPeval)];

choose_vector_ddP:=[choose_vector_ddP_Bernoulli[1],
choose_vector_ddP_Bernoulli[1]+choose_vector_ddP_Bernoulli
[2],choose_vector_ddP_Bernoulli[1]+
choose_vector_ddP_Bernoulli[2]+choose_vector_ddP_Bernoulli
[3]];
```

$$choose\_vector\_ddP\_Bernoulli := [0.956678641721771978534859076070487624517 8, 0.000149866272821146013105562104793584595732 1, 0.043021625732585729438929799719925206290 54, 0.000149866272821146013105562104793584595732 1]$$

(1.6.7.4)

choose_vector_ddP := [0.95667864172177197853485907607048762451 78, 0.95682850799459312454796463817528120911 35, 0.99985013372717885398689443789520641540 40] (1.6.7.4)

```
> choose_vector_ddH_Bernoulli:=[dxxKeval/ddHeval,2*dDeval*
  dxyKeval/ddHeval,ddDeval*dyKeval/ddHeval,dDeval^2*
  dyyKeval/ddHeval];
  choose_vector_ddH:=[choose_vector_ddH_Bernoulli[1],
  choose_vector_ddH_Bernoulli[1]+choose_vector_ddH_Bernoulli
  [2],choose_vector_ddH_Bernoulli[1]+
  choose_vector_ddH_Bernoulli[2]+choose_vector_ddH_Bernoulli
  [3]];
```

choose_vector_ddH_Bernoulli := [0.55367764846423254486726170158569384714 06, 0.33958532319106792598250069983729195832 00, 0.05472278194789704700546334525302004462 210, 0.05201424639680248214477425332399414991 731]

choose_vector_ddH := [0.55367764846423254486726170158569384714 06, 0.89326297165530047084976240142298580546 06, 0.94798575360319751785522574667600585008 27] (1.6.7.5)

```
> ch_ddP_or_ddH:=[evalf(ddPeval/(ddPeval+ddHeval))];
```

ch_ddP_or_ddH := [0.56734007644994804777628233685367379543 36] (1.6.7.6)

```
> ch_ddS_or_ddH:=[evalf(ddSeval/(ddSeval+ddHeval))];
```

ch_ddS_or_ddH := [0.16531998954096100636174291723441634703 86] (1.6.7.7)

All together

```
> ch_1_or_u;
  ch_1_or_v;
  ch_u_or_v;
  ch_dxu_or_dxv;
  choose_vector_dxu;
  choose_vector_dxv;
  ch_dyu_or_dyv;
  choose_vector_dyv;
  choose_vector_dyu;
  ch_K_in_dyK;
  ch_dxK_in_dxyK;
  ch_b_or_dxb;
  ch_3b_or_dyb;
  choose_vector_non_trivial_D;
  choose_vector_D;
  choose_vector_P;
  ch_y_or_P_or_H;
  ch_S_or_H;
  poisson_S_plus_H;
  poisson_at_least_1_S_plus_H;
  poisson_at_least_2_S_plus_H;
  choose_vector_dD;
  choose_vector_dS;
  choose_vector_dP;
  choose_vector_dH;
  ch_dP_or_dH;
  ch_dS_or_dH;
  choose_vector_ddD;
  choose_vector_ddS;
  choose_vector_ddP;
```

choose_vector_ddH;
ch_ddP_or_ddH;
ch_ddS_or_ddH;

[0.65295251033681916726502365056303868289 34]
[0.28039522499486659558475591879220540105 92]
[0.17156946157746118654760754670349873172 19]
[0.22980752953374312882718067053332124431 82]
[0.00105187713932158383879344925179959133 6700,
0.50052593856966079191939672462589979566 84]
[0.5]
[0.22969761559843038913952606839667943602 21]
[0.00062090800820379697680009889039482160 31175,
0.50031045400410189848840004944519741080 14]
[0.00043123688955594504460993276563503883 16594,
0.50021561844477797252230496638281751941 61]
[0.09736382064286582607785119655750788825 036]
[0.00034205213387951785063590145517007616 78339]
[0.00140694727476481448795333148854065549 3055]
[0.00172902829670659371928471391190777038 5558]
[0.46603904634839589009660545863589332516 95,
0.97745390878868452362176733699506166260 84]
[0.91393095083962268965377078684537679053 53,
0.95404248843043813535957152478366227580 83,
0.99805947936715893744185201731171185488 51]
[0.97767983827098610986898981012310840309 50]
[0.95212202941348519033925565531451712737 71,
0.99797838945998930993466066582471046557 92]
[0.95385432450130545561815998820163951601 76]
[0.95503022242123224730630308451852882409 03,
0.99897335234993598010573040983860754831 63,
0.99998431442348973099241972256458858556 76,
0.99999981998436386281590366560629093870 42,
0.99999999834596439467664147542376092233 53,
0.99999999998732962112831902216595813483 41,
0.99999999999991678394144334130431975152 23,
0.99999999999999952168672171717158138192 36,
0.99999999999999997555888658054660065043 9,
0.99999999999999999988758801965142738233 1,
0.99999999999999999999952995229779439622 2,
0.99999999999999999999999819820202809155 8,
0.99999999999999999999999993624319908235,
0.99999999999999999999999999979050341322,
0.99999999999999999999999999999935749563,
0.99999999999999999999999999999999815264,
0.99999999999999999999999999999999999502,
1.00000000000000000000000000000000000000,
1.00000000000000000000000000000000000000,
1.00000000000000000000000000000000000000,
1.00000000000000000000000000000000000000]
[0.97717027511942273294967624328166332513 17,
0.99965119737399646665816127143952976044 11,
0.99999599696405387209329853118118428581 65,

0.9999999632189507180417352316170128043908,
0.9999999971824679698519942158230440394 87,
0.9999999999981495114488636313328029814647,
0.9999999999999893636725811011893283868903,
0.9999999999999999456499125959805851510973,
0.9999999999999999997500277154992037334195,
0.9999999999999999999895474754932402787 26,
0.9999999999999999999999595933135787641720,
0.9999999999999999999999999858223001423642,
0.9999999999999999999999999999534139152886,
0.9999999999999999999999999999998571252956,
0.9999999999999999999999999999999995891998,
0.9999999999999999999999999999999999988930,
1.0000000000000000000000000000000000000002,
1.0000000000000000000000000000000000000005,
1.0000000000000000000000000000000000000005,
1.0000000000000000000000000000000000000005,
1.0000000000000000000000000000000000000005]
[0.984721558063965876371578449131034052 8525,
0.9998246568468491028580167921106432029449,
0.9999983888965165213051917991877382138919,
0.9999999876584932806392963752062780416867,
0.9999999991894389613469675552183181340 76,
0.9999999999995341018135550189337291383826,
0.9999999999999976193279731435321909073569,
0.9999999999999999890505783224105359338723,
0.9999999999999999999542152848471145434885,
0.9999999999999999999998244969467571390968,
0.9999999999999999999999937898069592209 50,
0.9999999999999999999999999979594110334049,
0.9999999999999999999999999999937417250766,
0.9999999999999999999999999999999820063301,
0.9999999999999999999999999999999999519330,
1.0000000000000000000000000000000000000505,
1.0000000000000000000000000000000000000629,
1.0000000000000000000000000000000000000629,
1.0000000000000000000000000000000000000629,
1.0000000000000000000000000000000000000629,
1.0000000000000000000000000000000000000629]
[0.385146067439818603410722680150780500 7197,
0.9076309562292024797575711378915160886748]
[0.066710627491838994078973066000735697 50144,
0.8958537007596913463391722043870576433116]
[0.9569654754198113448268853934226883952666]
[0.7639082430114071319914736581843454250145]
[0.8497707522399939853947748171909685807219]
[0.8065630979999361951674428436866237810075]
[0.078930428527852478895475841654043401 50041,
0.6014901096225815409521579129579569738547]
[0.4876368452244434788667479715333324837448,
0.4891906187921006823548138343659317306530,
0.4893857838633002021688201842121373689609,

```
    0.49181148083262799338827371390753175389951
[0.95667864172177197853485907607048762451780,
    0.95682850799459312454796463817528120911350,
    0.99985013372717885398689443789520641540400]
[0.55367764846423254486726170158569384714060,
    0.89326297165530047084976240142298580546060,
    0.94798575360319751785522574667600585008270]
                [0.56734007644994804777628233685367379543360]
                [0.16531998954096100636174291723441634703860]
```
(1.6.8.1)

2-connected planar graphs

Evaluation of the generating function of 2-connected planar graphs

```
> beta_1:=z*(6*x-2+x*z)/(4*x)+(1+z)*ln((1+y)/(1+z))-ln(1+z)/2+
  ln(1+x*z)/(2*x^2):
  beta_2:=(2*(1+x)*(1+w)*(z+w^2)+3*(w-z))/(2*(1+w)^2)-1/(2*x)*
  ln(1+x*z+x*w+x*w^2)+(1-4*x)/(2*x)*ln(1+w)+(1-4*x+2*x^2)/(4*x)
  *ln((1-x+x*z-x*w+x*w^2)/((1-x)*(z+w^2+1+w))):
  B:=subs(z=D,w=D*(1+u),x^2/2*beta_1-x/4*beta_2);
```

(2.1.1)

$$B := \frac{x^2\left(\frac{D\,(D\,x+6\,x-2)}{4\,x}+(1+D)\ln\left(\frac{1+y}{1+D}\right)-\frac{\ln(1+D)}{2}+\frac{\ln(D\,x+1)}{2\,x^2}\right)}{2}$$

$$-\frac{1}{4}\left(x\left(\frac{1}{2\,(1+D\,(1+u))^2}\left(2\,(1+x)\,(1+D\,(1+u))\,\left(D^2\,(1+u)^2\right.\right.\right.\right.$$

$$\left.\left.+D\right)+3\,D\,(1+u)-3\,D\right)-\frac{\ln\left(x\,D^2\,(1+u)^2+D\,x+x\,D\,(1+u)+1\right)}{2\,x}$$

$$+\frac{(1-4\,x)\ln(1+D\,(1+u))}{2\,x}$$

$$\left.\left.+\frac{\left(2\,x^2-4\,x+1\right)\ln\left(\frac{x\,D^2\,(1+u)^2+D\,x-x\,D\,(1+u)-x+1}{(1-x)\,\left(D^2\,(1+u)^2+D+D\,(1+u)+1\right)}\right)}{4\,x}\right)\right)$$

```
> system_network:={
  D=y+S+P+H,
  S=(y+P+H)*x*D,
  P=y*(exp(S+H)-1)+(exp(S+H)-S-H-1),
  H=K,

  u=x*D*(1+v)^2,
  v=D*(1+u)^2,
  K=1/2*D*(1/(1+x*D)+1/(1+D)-1-(1+u)^2*(1+v)^2/(1+u+v)^3)
  }:
  eval_network:=proc(x0,y0)
  global system_network:
  fsolve(subs({x=x0,y=y0},system_network),{u,v,K,S,P,H,D});
```

```
  end proc:
> eval_2connected:=proc(x0,y0)

  global system_network,B:
  local solutions_network:

  solutions_network:=fsolve(subs({x=x0,y=y0},system_network),
  {u,v,K,S,P,H,D}):
  evalf(subs({x=x0,y=y0,op(solutions_network)},B));
  end proc:
> B0=eval_2connected(.038191097669411,1);
```

$$B0 = 0.00073969957112330039054951543726406054544 \qquad (2.1.2)$$

Evaluation of the generating function of derivate 2-connected planar graphs

```
> dBx:=subs({DISS(x,y)=D,diff(DISS(x,y),x)=dD,u(x,DISS(x,y))=
  u,D[1](u)(x,DISS(x,y))=dxu,D[2](u)(x,DISS(x,y))=dyu},diff
  (subs({D=DISS(x,y),u=u(x,DISS(x,y))},B),x)):
> eval_derivate_2connected:=proc(x0,y0)

  local solutions_network, solutions_derivate_network,
  solutions_derivate_3_connected,
  solutions_derivate_binary_tree:

  global dBx, system_network, system_derivate_network,
  system_bi_derivate_network, system_derivate_3_connected,
  system_bi_derivate_3_connected, system_derivate_binary_tree:

  solutions_network:=fsolve(subs({x=x0,y=y0},system_network),
  {u,v,K,S,P,H,D}):

  solutions_derivate_binary_tree:=evalf(subs({x=x0,y=y0,op
  (solutions_network)},system_derivate_binary_tree)):

  solutions_derivate_3_connected:=evalf(subs({x=x0,y=y0,op
  (solutions_network)},system_derivate_3_connected)):


  solutions_derivate_network:=evalf(subs({x=x0,y=y0,op
  (solutions_network),op(solutions_derivate_3_connected)},
  system_derivate_network)):

  evalf(subs({x=x0,y=y0,op(solutions_network),op
  (solutions_derivate_network),op
  (solutions_derivate_binary_tree)},dBx));
  end proc:
> eval_derivate_2connected(.038191097669411,1);
```

$$0.039051802824590761935507953561299040192 3 \qquad (2.2.1)$$

```
> B2=-%*.038191097669411;
```

$$B2 = -0.00149143121584052615499501853674138179141 4 \qquad (2.2.2)$$

Evaluation of the generating function of bi-derivate 2-connected planar graphs

```
> dBxx:=subs({DISS(x,y)=D,diff(DISS(x,y),x)=dD,diff(DISS(x,y),
  x,x)=ddD,u(x,DISS(x,y))=u,D[1](u)(x,DISS(x,y))=dxu,D[2](u)(x,
  DISS(x,y))=dyu,D[1,1](u)(x,DISS(x,y))=dxxu,D[1,2](u)(x,DISS
```

```
(x,y))=dxyu,D[2,2](u)(x,DISS(x,y))=dyyu},diff(subs({D=DISS(x,
y),dD=diff(DISS(x,y),x),u=u(x,DISS(x,y)),dxu=D[1](u)(x,DISS
(x,y)),dyu=D[2](u)(x,DISS(x,y))},dBx),x)):
> eval_bi_derive_2connected:=proc(x0,y0)

local solutions_network, solutions_derivate_network,
solutions_bi_derivate_network,
solutions_derivate_binary_tree,
solutions_bi_derivate_binary_tree,
solutions_derivate_3_connected,
solutions_bi_derivate_3_connected:

global dBxx, system_network, system_derivate_network,
system_bi_derivate_network, system_derivate_binary_tree,
system_bi_derivate_binary_tree,system_derivate_3_connected,
system_bi_derivate_3_connected:

solutions_network:=fsolve(subs({x=x0,y=y0},system_network),
{u,v,K,S,P,H,D}):

solutions_derivate_binary_tree:=evalf(subs({x=x0,y=y0,op
(solutions_network)},system_derivate_binary_tree)):
solutions_bi_derivate_binary_tree:=evalf(subs({x=x0,y=y0,op
(solutions_network)},system_bi_derivate_binary_tree)):

solutions_derivate_3_connected:=evalf(subs({x=x0,y=y0,op
(solutions_network),op(solutions_derivate_binary_tree)},
system_derivate_3_connected)):
solutions_bi_derivate_3_connected:=evalf(subs({x=x0,y=y0,op
(solutions_network),op(solutions_derivate_binary_tree),op
(solutions_bi_derivate_binary_tree)},
system_bi_derivate_3_connected)):


solutions_derivate_network:=evalf(subs({x=x0,y=y0,op
(solutions_network),op(solutions_derivate_3_connected)},
system_derivate_network)):
solutions_bi_derivate_network:=evalf(subs({x=x0,y=y0,op
(solutions_network),op(solutions_derivate_network),op
(solutions_derivate_3_connected),op
(solutions_bi_derivate_3_connected)},
system_bi_derivate_network)):


evalf(subs({x=x0,y=y0,op(solutions_network),op
(solutions_derivate_network),op
(solutions_bi_derivate_network),op
(solutions_derivate_binary_tree),op
(solutions_bi_derivate_binary_tree)},dBxx));


end proc:
> eval_bi_derive_2connected(.038191097669411,1);
```

$$1.051966755476522130558698949519598978056 \qquad (2.3.1)$$

```
> B4=.038191097669411^2*%/2;
```

$$B4 = 0.000767178284503197446082025173752457361780 7 \qquad (2.3.2)$$

Evaluation of the generating function of tri-derivate 2-connected planar graphs

```
> dBxxx:=subs({DISS(x,y)=D,diff(DISS(x,y),x)=dD,diff(DISS(x,
  y),x,x)=ddD,diff(DISS(x,y),x,x,x)=dddD,u(x,DISS(x,y))=u,D[1]
  (u)(x,DISS(x,y))=dxu,D[2](u)(x,DISS(x,y))=dyu,D[1,1](u)(x,
  DISS(x,y))=dxxu,D[1,2](u)(x,DISS(x,y))=dxyu,D[2,2](u)(x,DISS
  (x,y))=dyyu,D[1,1,1](u)(x,DISS(x,y))=dxxxu,D[1,1,2](u)(x,DISS
  (x,y))=dxxyu,D[1,2,2](u)(x,DISS(x,y))=dxyyu,D[2,2,2](u)(x,
  DISS(x,y))=dyyyu},diff(subs({D=DISS(x,y),dD=diff(DISS(x,y),
  x),ddD=diff(DISS(x,y),x,x),u=u(x,DISS(x,y)),dxu=D[1](u)(x,
  DISS(x,y)),dyu=D[2](u)(x,DISS(x,y)),dxxu=D[1,1](u)(x,DISS(x,
  y)),dxyu=D[1,2](u)(x,DISS(x,y)),dyyu=D[2,2](u)(x,DISS(x,y))},
  dBxx),x)):
> eval_tri_derivate_2connected:=proc(x0,y0)

  local solutions_network, solutions_derivate_network,
  solutions_bi_derivate_network,
  solutions_tri_derivate_network,
  solutions_derivate_binary_tree,
  solutions_bi_derivate_binary_tree,
  solutions_tri_derivate_binary_tree,
  solutions_derivate_3_connected,
  solutions_bi_derivate_3_connected,
  solutions_tri_derivate_3_connected:

  global dBxxx, system_network, system_derivate_network,
  system_bi_derivate_network, system_tri_derivate_network,
  system_derivate_binary_tree, system_bi_derivate_binary_tree,
  system_tri_derivate_binary_tree,system_derivate_3_connected,
  system_bi_derivate_3_connected,
  system_tri_derivate_3_connected:

  solutions_network:=fsolve(subs({x=x0,y=y0},system_network),
  {u,v,K,S,P,H,D}):

  solutions_derivate_binary_tree:=evalf(subs({x=x0,y=y0,op
  (solutions_network)},system_derivate_binary_tree)):
  solutions_bi_derivate_binary_tree:=evalf(subs({x=x0,y=y0,op
  (solutions_network)},system_bi_derivate_binary_tree)):
  solutions_tri_derivate_binary_tree:=evalf(subs({x=x0,y=y0,op
  (solutions_network)},system_tri_derivate_binary_tree)):

  solutions_derivate_3_connected:=evalf(subs({x=x0,y=y0,op
  (solutions_network),op(solutions_derivate_binary_tree)},
  system_derivate_3_connected)):
  solutions_bi_derivate_3_connected:=evalf(subs({x=x0,y=y0,op
  (solutions_network),op(solutions_derivate_binary_tree),op
  (solutions_bi_derivate_binary_tree)},
  system_bi_derivate_3_connected)):
  solutions_tri_derivate_3_connected:=evalf(subs({x=x0,y=y0,op
  (solutions_network),op(solutions_derivate_binary_tree),op
  (solutions_bi_derivate_binary_tree),op
  (solutions_tri_derivate_binary_tree)},
  system_tri_derivate_3_connected)):
```

```
solutions_derivate_network:=evalf(subs({x=x0,y=y0,op
(solutions_network),op(solutions_derivate_3_connected)},
system_derivate_network)):
solutions_bi_derivate_network:=evalf(subs({x=x0,y=y0,op
(solutions_network),op(solutions_derivate_network),op
(solutions_derivate_3_connected),op
(solutions_bi_derivate_3_connected)},
system_bi_derivate_network)):
solutions_tri_derivate_network:=evalf(subs({x=x0,y=y0,op
(solutions_network),op(solutions_derivate_network),op
(solutions_bi_derivate_network),op
(solutions_derivate_3_connected),op
(solutions_bi_derivate_3_connected),op
(solutions_tri_derivate_3_connected)},
system_tri_derivate_network)):


evalf(subs({x=x0,y=y0,op(solutions_network),op
(solutions_derivate_network),op
(solutions_bi_derivate_network),op
(solutions_tri_derivate_network),op
(solutions_derivate_binary_tree),op
(solutions_bi_derivate_binary_tree),op
(solutions_tri_derivate_binary_tree)},dBxxx));


end proc:
> eval_tri_derivate_2connected(.038191097669411,1);
```

$$1.25782199316836082772369254\ 10^{6} \tag{2.4.1}$$

```
> solutions_singularity:=find_singularity(1):R:=subs(%,xsi);
```

$$R := 0.038191097669411335391152564045422359553 88 \tag{2.4.2}$$

```
> eval_tri_derivate_2connected(.038191097669411,1);
```

$$1.25782199316836082772369254\ 10^{6} \tag{2.4.3}$$

```
> B5=-(.038191097669411^3*%*8)*(1-.038191097669411/R)^(1/2)
  /15;
```

$$B5 = -3.50186230286349748814835997323955077541 3\ 10^{-6} \tag{2.4.4}$$

```
> eval_tri_derivate_2connected(.038191097669,1);dx:=
  0.00000000000000001;evalf((eval_bi_derivate_2connected
  (.038191097669+dx,1)-eval_bi_derivate_2connected
  (.038191097669,1))/dx);
```

$$35918.358393716869438572377126800000000 01$$

$$dx := 1.\ 10^{-17}$$

$$35918.576697360712506334965900000000000 00 \tag{2.4.5}$$

Connected planar graphs

Evaluation of the generating function of pointed connected planar graphs

```
> inverse_F:=proc(x,y0)
  evalf(x*exp(-eval_derivate_2connected(x,y0)));
```

```
   end proc;
```
$$inverse\_F := \mathbf{proc}(x, y0) \quad \textbf{(3.1.1)}$$
$$evalf(x * \exp(-eval\_derivate\_2connected(x, y0)))$$
```
end proc
> eval_F:=proc(z,y0)
  local inverse_F_at_z:
  inverse_F_at_z:=x-> evalf(inverse_F(x,y0)-z):
  fsolve(inverse_F_at_z):
  end proc:

> eval_F(.03672841258183,1);
```
0.038191097669402429946807661968678134798 50 **(3.1.2)**
```
> inverse_F(.038191097669402429946807661968678134798 50,1);
```
0.036728412581830000000000000000000000506352 **(3.1.3)**

Evaluation of the generating function of connected planar graphs

```
> eval_C:=proc(z,y)
  local value_F:

  value_F:=eval_F(z,y):
  value_F*log(z)-value_F*log(value_F)+value_F+eval_2connected
  (value_F,y):
  end proc;
```
$$eval\_C := \mathbf{proc}(z, y) \quad \textbf{(3.2.1)}$$
$$\mathbf{local}\ value\_F;$$
$$value\_F := eval\_F(z, y);$$
$$value\_F * \log(z) - value\_F * \log(value\_F) + value\_F + eval\_2connected(value\_F, y)$$
```
end proc
```

Evaluation of the generating function of bi-derivate connected planar graphs

```
> eval_derivate_inverse_F:=proc(x0,y0)
     evalf(exp(-eval_derivate_2connected(x0,y0))*(1-x0*
  eval_bi_derivate_2connected(x0,y0))):
  end proc:
> eval_bi_derivate_connected:=proc(z0,y0)
  local F,dC,dF:

  F:=eval_F(z0,y0):
  dF:=evalf(1/eval_derivate_inverse_F(F,y0)):
  evalf(-F/z0^2+dF/z0):

  end proc:
```

Evaluation of the generating function of tri-derivate connected planar graphs

```
> eval_bi_derivate_inverse_F:=proc(x0,y0)
     evalf(exp(-eval_derivate_2connected(x0,y0))*(-2*
  eval_bi_derivate_2connected(x0,y0)+x0*
  eval_bi_derivate_2connected(x0,y0)^2-x0*
  eval_tri_derivate_2connected(x0,y0))):
```

```
  end proc:
> eval_tri_derivate_connected:=proc(z0,y0)
  local F,dC,ddF,dF:

  F:=eval_F(z0,y0):
  dF:=evalf(1/eval_derivate_inverse_F(F,y0)):
  ddF:=evalf(-eval_bi_derivate_inverse_F(F,y0)
  /eval_derivate_inverse_F(F,y0)^3):
  evalf(ddF/z0-2*dF/z0^2+2*F/z0^3):

  end proc:
```

Planar graphs and connected planar graphs together

We want to make the expensive calculation of the inverse just once, so we derive from one evaluation of F all the generating functions of connected and planar graphs

```
> evaluate_planar_and_connected:=proc(z0,F,y0)
  local dF,ddF,Ceval,dCeval,ddCeval,dddCeval,Geval,dGeval,
  ddGeval,dddGeval:
  dF:=evalf(1/eval_derivate_inverse_F(F,y0)):
  ddF:=evalf(-eval_bi_derivate_inverse_F(F,y0)
  /eval_derivate_inverse_F(F,y0)^3):
  dCeval:=evalf(F/z0):
  Ceval:=F*log(z0)-F*log(F)+F+eval_2connected(F,y0):
  ddCeval:=evalf(-F/z0^2+dF/z0):
  dddCeval:=evalf(ddF/z0-2*dF/z0^2+2*F/z0^3):
  Geval:=evalf(exp(Ceval)):
  dGeval:=evalf(dCeval*Geval):
  ddGeval:=evalf(ddCeval*Geval+dCeval*dGeval):
  dddGeval:=evalf(dddCeval*Geval+2*ddCeval*dGeval+dCeval*ddGeval)
  :
  {C=Ceval,dC=dCeval,ddC=ddCeval,dddC=dddCeval,G=Geval,dG=dGeval,
  ddG=ddGeval,dddG=dddGeval}:
  end proc:
> evaluate_planar_and_connected(.03672841258183,
  .038191097669402429946807661968678167979 91,1);
```

$$\{C = 0.037439366024685548490805435227680414199 53,\ G = 1.038149048069666547989910110165655146797,\ dC = 1.039824348093280741233629384704015601806,\ dG = 1.079492657132700989555944229432597122144,\ ddC = 1.185032516943684908500357051427308908 74,\ ddG = 2.352723127871181713240436977837925598242,\ dddC = 310370.293284021647870450441344793920805 5,\ dddG = 322215.629408503985297581172298887699 2872\} \quad (4.1)$$

Calculating all the choose-vectors

Choice of the number of vertices and possibly of balance edges-vertices

N is the wanted average number of vertices

```
> N:=1000; mu:=2; y0:=1;
```

$$N := 1000$$

$$\mu := 2$$

$$y0 := 1 \tag{5.1.1}$$

Finding the good z from singularity of generating functions of planar graphs

```
> h_t:=t^2*(1-t)*(18+36*t+5*t^2)/(2*(3+t)*(1+2*t)*(1+3*t)^2):
  y0_t:=(1+2*t)/((1+3*t)*(1-t))*exp(-h_t)-1:
  rho:=-1/16*sqrt(1+3*t)*(-1+t)^3/t^3*exp(1/16*ln(1+t)*(3*t-1)*
  (1+t)^3/t^3-1/32*ln(1+2*t)*(1+3*t)*(-1+t)^3/t^3-1/64*(-1+t)*
  (185*t^4+698*t^3-217*t^2-160*t+6)/t/(1+3*t)^2/(3+t)):
> find_singularity_connected:=proc(y0)
  evalf(subs(t=fsolve(y0_t=y0,t), rho));
  end proc;
```

$$\begin{aligned} &find\_singularity\_connected := \mathbf{proc}(y0) \\ &\quad evalf(subs(t = fsolve(y0\_t = y0, t), \mathrm{rho})) \\ &\mathbf{end\ proc} \end{aligned} \tag{5.2.1}$$

```
> Rc:=find_singularity_connected(1);
```

$$Rc := 0.036728412581838220293476617184035408144 72 \tag{5.2.2}$$

```
> 1/%;
```

$$27.226877768588576467079458051494458287 52 \tag{5.2.3}$$

```
> z0:=evalf(Rc*(1-1/(2*N)));
```

$$z0 := 0.036710048375547301183329878875443390440 65 \tag{5.2.4}$$

If a particular balance edge-vertices is wanted, execute this part to find the good y from the balance edges-vertices

```
> rho_t:=-1/16*sqrt(1+3*t)*(-1+t)^3*t^(-3)*exp(A);
```

$$rho\_t := -\frac{\sqrt{1+3t}\,(-1+t)^3\,e^A}{16\,t^3} \tag{5.3.1}$$

```
> A:=log(1+t)*(3*t-1)*(1+t)^3/(16*t^3)-log(1+2*t)*(1+3*t)*(-1+
  t)^3/(32*t^3)-(-1+t)*(185*t^4+698*t^3-217*t^2-160*t+6)/(64*t*
  (1+3*t)^2*(3+t));
```

$$A := \frac{\ln(1+t)\,(3t-1)\,(1+t)^3}{16\,t^3} - \frac{\ln(1+2t)\,(1+3t)\,(-1+t)^3}{32\,t^3} - \frac{(-1+t)\left(185t^4+698t^3-217t^2-160t+6\right)}{64\,t\,(1+3t)^2\,(3+t)} \tag{5.3.2}$$

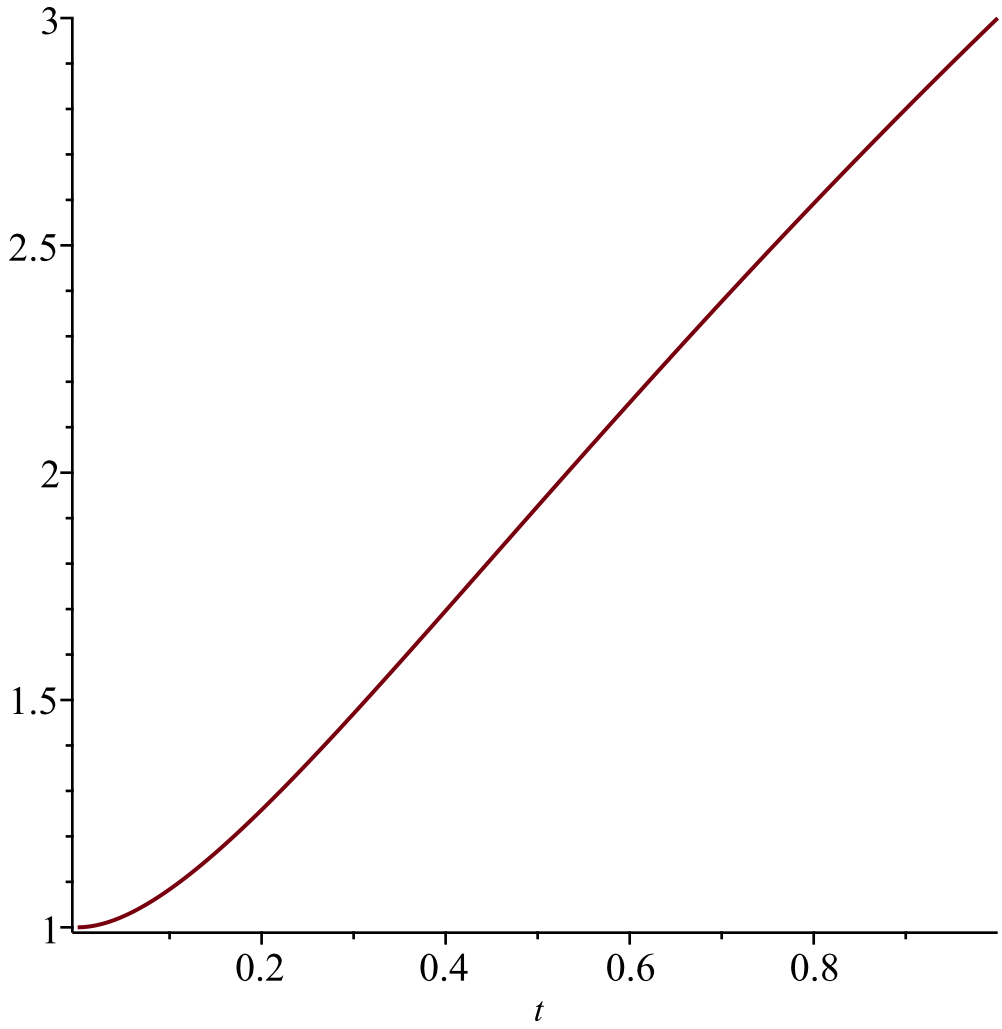
```
> Y:=(1+2*t)/(1+3*t)/(1-t)*exp(-t^2*(1-t)*(18+36*t+5*t^2)/(2*
  (3+t)*(1+2*t)*(1+3*t)^2))-1;
```

$$Y := \frac{(1+2t)\,e^{-\frac{t^2(1-t)\left(5t^2+36t+18\right)}{2(3+t)(1+2t)(1+3t)^2}}}{(1+3t)\,(1-t)} - 1 \tag{5.3.3}$$

```
> rho_subt_prime:=simplify(diff(rho_t, t)):
  Y_prime:=simplify(diff(Y, t)):
  rho_prime_t:=simplify( rho_subt_prime*1/Y_prime ):
  mu_t:=simplify(-Y*rho_prime_t/rho_t):
  t_mu:=mu->fsolve(mu_t=mu, t, 0..1):
  y_from_expected_edges:=expect_edges->evalf(subs(t=t_mu
  (expect_edges),Y)):
  y_from_expected_edges(1.1);
  y_from_expected_edges(2);
  y_from_expected_edges(2.2);
  y_from_expected_edges(2.9);
```

0.012966814282347086203536165759856380032
0.613408306229252191639801077529145495803
0.969701292577611404868130565290704156376
13.807868616660555822922039516667980510008 **(5.3.4)**

```
> plot(mu_t, t=0+0.0001..1-0.0001);
```




```
> y0:=y_from_expected_edges(mu);
```

$$y0 := 0.613408306229252191639801077529145495803 \quad \textbf{(5.3.5)}$$

```
> x0:=eval_F(z0,y0);
```

$$x0 := 0.037572423264915159508150505717898017965 51 \quad \textbf{(5.1)}$$

Calculating the choose-vectors of bicolored binary trees

```
> solutions_networks:=eval_network(x0,y0);
```

$$solutions\_networks := \{D = 0.63780665105457654378986811428105278222 64, H = 0.000082377383152041591768497805401560105638 82, K = 0.000082377383152041591768497805401560105638 82, P = 0.009389307794846823503367685024265925290141, S = 0.014926659647325487054930853922239801027 59, u = 0.071962869757198588831781155881309984095 86, v = 0.732906425337312534729034476087514845833 5\} \quad \textbf{(5.4.1)}$$

```
> u_eval:=subs(solutions_networks,u):v_eval:=subs
  (solutions_networks,v);
```

$$v\_eval := 0.73290642533731253472903447608751484583 \quad (5.4.2)$$

```
> ch_1_or_u:=[evalf(1/(1+u_eval))];
```

$$ch\_1\_or\_u := [0.93286813210843925863755845937409772383 21] \quad (5.4.3)$$

```
> ch_1_or_v:=[evalf(1/(1+v_eval))];
```

$$ch\_1\_or\_v := [0.57706520408645185403226336431748059815 25] \quad (5.4.4)$$

```
> ch_u_or_v:=[evalf(u_eval/(u_eval+v_eval))];
```

$$ch\_u\_or\_v := [0.08940938633862086548106554134360917655423] \quad (5.4.5)$$

Calculating the choose-vectors of pointed bicolored binary trees

```
> solution_derivate_binary_tree:=subs({x=x0,op
  (solutions_networks)},system_derivate_binary_tree);
```

$$solution\_derivate\_binary\_tree := \{dxu = 2.16070079470421020697685116058489171290 5, dxv = 2.95456408117725059123552387355704562676 8, dyu = 0.23495019484475095031567858620209041956 09, dyv = 1.47037766263287210402127056128695341162 4\} \quad (5.5.1)$$

Here pU and pV stand for the generating functions of black-rooted and white-rooted bicolored binary trees with a pointed black vertex

```
> dxu_eval:=evalf(subs(solution_derivate_binary_tree,dxu))
  :dxv_eval:=evalf(subs(solution_derivate_binary_tree,dxv))
  :dyu_eval:=evalf(subs(solution_derivate_binary_tree,dyu))
  :dyv_eval:=evalf(subs(solution_derivate_binary_tree,dyv)):
```

```
> ch_dxu_or_dxv:=[evalf(dxu_eval/(dxu_eval+dxv_eval))];
```

$$ch\_dxu\_or\_dxv := [0.42240252403975052044002916518615778799 79] \quad (5.5.2)$$

```
> D_eval:=subs(solutions_networks,D);
```

$$D\_eval := 0.63780665105457654378986811428105278222 64 \quad (5.5.3)$$

```
> choose_vector_dxu_Bernoulli:=[(1+v_eval)^2*D_eval/dxu_eval,
  x0*D_eval*dxv_eval*(1+v_eval)/dxu_eval,(1+v_eval)*x0*D_eval*
  dxv_eval/dxu_eval];
  choose_vector_dxu:=[choose_vector_dxu_Bernoulli[1],
  choose_vector_dxu_Bernoulli[1]+choose_vector_dxu_Bernoulli[2]
  ];
```

$$choose\_vector\_dxu\_Bernoulli := [0.88643038861594992869656568763581707186 79, 0.05678480569202503565171715618209146406648, 0.05678480569202503565171715618209146406648]$$

$$choose\_vector\_dxu := [0.88643038861594992869656568763581707186 79, 0.94321519430797496434828284381790853593 44] \quad (5.5.4)$$

```
> choose_vector_dxv:=[0.5];
```

$$choose\_vector\_dxv := [0.5] \quad (5.5.5)$$

```
> ch_dyu_or_dyv:=[evalf(dyu_eval/(dyu_eval+dyv_eval))];
```

$$ch\_dyu\_or\_dyv := [0.13777420794161507226803628047524164369 77] \quad (5.5.6)$$

```
> choose_vector_dyv_Bernoulli:=[(1+u_eval)^2/dyv_eval,dyu_eval*
  D_eval*(1+u_eval)/dyv_eval,(1+u_eval)*D_eval*
  dyu_eval/dyv_eval];
  choose_vector_dyv:=[choose_vector_dyv_Bernoulli[1],
  choose_vector_dyv_Bernoulli[1]+choose_vector_dyv_Bernoulli[2]
  ];
```

$$choose\_vector\_dyv\_Bernoulli := [0.78150289095149306345481443614191848565 87, 0.10924855452425346827259278192904075717 09, 0.10924855452425346827259278192904075717 09]$$

$$choose\_vector\_dyv := [0.78150289095149306345481443614191848565 87, 0.89075144547574653172740721807095924282 96] \quad (5.5.7)$$

```
> choose_vector_dyu_Bernoulli:=[x0*(1+v_eval)^2/dyu_eval,x0*
```

```
D_eval*dyv_eval*(1+v_eval)/dyu_eval,x0*D_eval*(1+v_eval)*
dyv_eval/dyu_eval];
choose_vector_dyu:=[choose_vector_dyu_Bernoulli[1],
choose_vector_dyu_Bernoulli[1]+choose_vector_dyu_Bernoulli[2]
];
```

$$choose\_vector\_dyu\_Bernoulli := [0.48022373440722587789190336828033360320 49,\ 0.25988813279638706105404831585983319839 76,\ 0.25988813279638706105404831585983319839 76]$$

$$choose\_vector\_dyu := [0.48022373440722587789190336828033360320 49,\ 0.74011186720361293894595168414016680160 25] \quad (5.5.8)$$

```
> ch_b_or_dxb:=[evalf((u_eval+v_eval)/(u_eval+v_eval+x0*
dxu_eval+x0*dxv_eval))];
```

$$ch\_b\_or\_dxb := [0.80724081351486246321389307171424202549 20] \quad (5.5.9)$$

```
> ch_3b_or_dyb:=[evalf((3*u_eval+3*v_eval)/(3*u_eval+3*v_eval+
D_eval*dyu_eval+D_eval*dyv_eval))];
```

$$ch\_3b\_or\_dyb := [0.68943937167547733763992078629138053384 96] \quad (5.5.10)$$

Calculating the choose-vectors of 3-connected networks

```
> solution_derivate_3_connected:=subs({x=x0,op(eval_network
(x0,y0))},system_derivate_3_connected);
solution_bi_derivate_3_connected:=subs({x=x0,op(eval_network
(x0,y0))},system_bi_derivate_3_connected);
```

$$solution\_derivate\_3\_connected := \{dxK = 0.00463118246588456114847814832532560426 8230,\ dyK = 0.00067917306357466825599792059911841052612 05\}$$

$$solution\_bi\_derivate\_3\_connected := \{dxxK = 0.14525691617753654702381747274670685673 47,\ dxyK = 0.03928066227449508126801960756239290835536,\ dyyK = 0.00468811527622347696103753301794425385687 7\} \quad (5.6.1)$$

```
> K_eval:=subs(solutions_networks,K);dxK_eval:=subs
(solution_derivate_3_connected,dxK);dyK_eval:=subs
(solution_derivate_3_connected,dyK);dxyK_eval:=subs
(solution_bi_derivate_3_connected,dxyK);
```

$$K\_eval := 0.00008237738315204159176849780540156010563882$$

$$dxK\_eval := 0.00463118246588456114847814832532560426 8230$$

$$dyK\_eval := 0.00067917306357466825599792059911841052612 05$$

$$dxyK\_eval := 0.03928066227449508126801960756239290835536 \quad (5.6.2)$$

```
> ch_K_in_dyK:=[evalf(K_eval/(y0*dyK_eval))];
```

$$ch\_K\_in\_dyK := [0.19773242203269807781218157736645052476 94] \quad (5.6.3)$$

```
> ch_dxK_in_dxyK:=[evalf(dxK_eval/(y0*dxyK_eval))];
```

$$ch\_dxK\_in\_dxyK := [0.19220444997055135398962461968550027288 30] \quad (5.6.4)$$

Calculating the choose-vectors of networks

We recall the system verified by the networks

```
> system_verified_by_network:={
D=y+S+P+H,
S=(y+P+H)*x*D,
P=y*(exp(S+H)-1)+(exp(S+H)-S-H-1),
H=K}:
> solutions_networks:=eval_network(x0,y0);
```

$$solutions\_networks := \{D = 0.63780665105457654378986811428105278222 64,\ H = 0.00008237738315204159176849780540156010563882,\ K = 0.00008237738315204159176849780540156010563882,\ P$$ (5.7.1)

$= 0.009389307794846823503367685024265925290141, S$
$= 0.014926659647325487054930853922239801027 59, u$
$= 0.071962869757198588831781155881309984095 86, v$
$= 0.732906425337312534729034476087514845833 5\}$

```
> Deval:=subs(solutions_networks,D):Seval:=subs
  (solutions_networks,S):Peval:=subs(solutions_networks,P)
  :Heval:=subs(solutions_networks,H):
> choose_vector_non_trivial_D_Bernoulli:=[evalf(Seval/(Deval-
  y0)),evalf(Peval/(Deval-y0)),evalf(Heval/(Deval-y0))];
  choose_vector_non_trivial_D:=
  [choose_vector_non_trivial_D_Bernoulli[1],
  choose_vector_non_trivial_D_Bernoulli[1]+
  choose_vector_non_trivial_D_Bernoulli[2]];
```

$$choose\_vector\_non\_trivial\_D\_Bernoulli := [0.611789846983075078441215944358404765 6820, 0.384833801721711744532685778620433186 1881, 0.003376351295213177026098277021162048 128595]$$

$$choose\_vector\_non\_trivial\_D := [0.611789846983075078441215944358404765 6820, 0.996623648704786822973901722978837951 8701] \quad (5.7.2)$$

```
> choose_vector_D_Bernoulli:=[evalf(y0/Deval),evalf
  (Seval/Deval),evalf(Peval/Deval),evalf(Heval/Deval)];
  choose_vector_D:=[choose_vector_D_Bernoulli[1],
  choose_vector_D_Bernoulli[1]+choose_vector_D_Bernoulli[2],
  choose_vector_D_Bernoulli[1]+choose_vector_D_Bernoulli[2]+
  choose_vector_D_Bernoulli[3]];
```

$$choose\_vector\_D\_Bernoulli := [0.961746487301468093911176015311632554 6148, 0.023403110680399954248929391707670974 69375, 0.014721244720985809924097888027383718 00991, 0.000129157297146141915796704953312752 6815044]$$

$$choose\_vector\_D := [0.961746487301468093911176015311632554 6148, 0.985149597981868048160105407019303529 3086, 0.999870842702853858084203295046687247 3185] \quad (5.7.3)$$

```
> choose_vector_P:=[evalf(y0*(exp(Seval+Heval)-1))/Peval];
```

$$choose\_vector\_P := [0.987943601714935769465858718332140733514 5] \quad (5.7.4)$$

```
> ch_y_or_P_or_H_Bernoulli:=[evalf(y0/(y0+Peval+Heval)),evalf
  (Peval/(y0+Peval+Heval)),evalf(Heval/(y0+Peval+Heval))];
  ch_y_or_P_or_H:=[ch_y_or_P_or_H_Bernoulli[1],
  ch_y_or_P_or_H_Bernoulli[1]+ch_y_or_P_or_H_Bernoulli[2]];
```

$$ch\_y\_or\_P\_or\_H\_Bernoulli := [0.984793723817328251517490460853723531 3935, 0.015074023767618361969115727746978205 14993, 0.000132252415053386513393811399298263 4566907]$$

$$ch\_y\_or\_P\_or\_H := [0.984793723817328251517490460853723531 3935, 0.999867747584946613486606188600701736 5434] \quad (5.7.5)$$

```
> ch_S_or_H:=[evalf(Seval/(Seval+Heval))];
```

$$ch\_S\_or\_H := [0.994511481117358488736163364208744385 0972] \quad (5.7.6)$$

We compute the vector for the Poisson laws of parameter S+H. Notice that the 1+... is only used to have an output without exponent notation. When put in the file, the first 1 has to be replaced by a 0.

```
> exp_S_plus_H:=evalf(exp(Seval+Heval)):
  poisson_S_plus_H:=[seq(0,i=1..21)]:
  poisson_S_plus_H[1]:=evalf(1/exp_S_plus_H):
  for i from 2 to 21 do poisson_S_plus_H[i]:=poisson_S_plus_H
```

```
  [i-1]+evalf((Seval+Heval)^(i-1)/(i-1)!/exp_S_plus_H);
  od:
  poisson_S_plus_H;
```

[0.985103037156666591528506491105006761274l,
0.999888485120186881232314573528863482898 2,
0.999999442788185218528453396914376475030 3,
0.999999997910767819353760469527106435240 8,
0.999999999993731669031287389036003077087 2,
0.999999999999984325341878595036401285176 3,
0.999999999999999966400229347105494386456 3,
0.999999999999999999936975630290863928550 0,
0.999999999999999999999894913646621282909 9,
0.999999999999999999999998422970454685272,
0.999999999999999999999999997848454459906,
0.999999999999999999999999999997309206915,
0.999999999999999999999999999999996893621,
0.999999999999999999999999999999999996666,
0.999999999999999999999999999999999999993,
0.999999999999999999999999999999999999996,
0.999999999999999999999999999999999999996,
0.999999999999999999999999999999999999996,
0.999999999999999999999999999999999999996,
0.999999999999999999999999999999999999996,
0.999999999999999999999999999999999999996]

(5.7.7)

```
> exp_at_least_1_S_plus_H:=evalf(exp(Seval+Heval)-1):
  poisson_at_least_1_S_plus_H:=[seq(0,i=1..21)]:
  poisson_at_least_1_S_plus_H[1]:=evalf((Seval+Heval)
  /exp_at_least_1_S_plus_H):
  for i from 2 to 21 do
  poisson_at_least_1_S_plus_H[i]:=poisson_at_least_1_S_plus_H
  [i-1]+evalf((Seval+Heval)^i/i!/exp_at_least_1_S_plus_H);
  od:
  poisson_at_least_1_S_plus_H;
```

[0.992514254013661370011236683099122184744 5,
0.999962595609545281819700754714375285602 9,
0.999999859754488037727833064198521995399 5,
0.999999999579220876454171092706998855870 0,
0.999999999989477950447852808977501194517,
0.999999999999977445220877401291528260160,
0.999999999999999957693141634343052813727,
0.999999999999999999929457867027073475742,
0.999999999999999999999894137512330543891,
0.999999999999999999999999855571530739365,
0.999999999999999999999999999819373041724,
0.999999999999999999999999999999791475614,
0.999999999999999999999999999999999776149,
0.999999999999999999999999999999999999463,
0.999999999999999999999999999999999999686,
0.999999999999999999999999999999999999686,
0.999999999999999999999999999999999999686,
0.999999999999999999999999999999999999686,
0.999999999999999999999999999999999999686,

(5.7.8)

```
   0.99999999999999999999999999999999999999686]
> exp_at_least_2_S_plus_H:=evalf(exp(Seval+Heval)-Seval-
  Heval-1):
  poisson_at_least_2_S_plus_H:=[seq(0,i=1..21)]:
  poisson_at_least_2_S_plus_H[1]:=evalf((Seval+Heval)
  ^2/2/exp_at_least_2_S_plus_H):
  for i from 2 to 21 do
  poisson_at_least_2_S_plus_H[i]:=poisson_at_least_2_S_plus_H
  [i-1]+evalf((Seval+Heval)^(i+1)/(i+1)!
  /exp_at_least_2_S_plus_H);
  od:
  poisson_at_least_2_S_plus_H;
```

$$[0.995003251443078537371096704643804702489 5,\ 0.999981264991863440458004737102017304682 5,\ 0.999999943789286423324507128666882717710 5,\ 0.999999999859438864592122577728868196659 1,\ 0.999999999999698696974706851800580912613 6,\ 0.999999999999999434834438105884092618100 4,\ 0.999999999999999999057647252502758984928 6,\ 0.999999999999999999985858124512553191829,\ 0.999999999999999999999980706202224373320,\ 0.999999999999999999999999975870546756835,\ 0.999999999999999999999999999972143874808,\ 0.999999999999999999999999999999970164319,\ 0.999999999999999999999999999999999996163,\ 1.000000000000000000000000000000000002601,\ 1.000000000000000000000000000000000002604,\ 1.000000000000000000000000000000000002604,\ 1.000000000000000000000000000000000002604,\ 1.000000000000000000000000000000000002604,\ 1.000000000000000000000000000000000002604,\ 1.000000000000000000000000000000000002604]$$ **(5.7.9)**

Calculating the choose-vectors of derivate networks

We recall the system verified by pointed networks

```
> equations_derivate_network:={
  dD=dS+dP+dH,
  dS=(dP+dH)*x*D+(y+P+H)*D+(y+P+H)*x*dD,
  dP=y*(dS+dH)*exp(S+H)+(dS+dH)*(exp(S+H)-1),
  dH=dxK+dD*dyK
  }:
> solution_derivate_network:=eval_derivate_network(x0,y0);
```

$$solution\_derivate\_network := \{dD = 0.69656900427265003455325850190118784479 66,\ dH = 0.00510427337050757305444031263634611369204 7,\ dP = 0.27126299893671352617907930754674322390 02,\ dS = 0.42020173196542893531973888171809850720 44\}$$ **(5.8.1)**

```
> dDeval:=subs(solution_derivate_network,dD):dSeval:=subs
  (solution_derivate_network,dS):dPeval:=subs
  (solution_derivate_network,dP):dHeval:=subs
  (solution_derivate_network,dH):dxKeval:=subs
  (solution_derivate_3_connected,dxK):dyKeval:=subs
```

```
(solution_derivate_3_connected,dyK):
> choose_vector_dD_Bernoulli:=[evalf(dSeval/dDeval),evalf
(dPeval/dDeval),evalf(dHeval/dDeval)];
choose_vector_dD:=[choose_vector_dD_Bernoulli[1],
choose_vector_dD_Bernoulli[1]+choose_vector_dD_Bernoulli[2]];
```

$$choose\_vector\_dD\_Bernoulli := [0.60324494685806343992132227587783848516 48,\ 0.38942731771414875438289509790656917450 34,\ 0.00732773542778780569578262621559234033 1900]$$

$$choose\_vector\_dD := [0.60324494685806343992132227587783848516 48,\ 0.99267226457221219430421737378440765966 82] \quad (5.8.2)$$

```
> choose_vector_dS_Bernoulli:=[evalf((dPeval+dHeval)*x0*
Deval/dSeval),evalf((y0+Peval+Heval)*Deval/dSeval),evalf((y0+
Peval+Heval)*x0*dDeval/dSeval)];
choose_vector_dS:=[choose_vector_dS_Bernoulli[1],
choose_vector_dS_Bernoulli[1]+choose_vector_dS_Bernoulli[2]];
```

$$choose\_vector\_dS\_Bernoulli := [0.01576111812428889709138079692984191481 274,\ 0.94544351226293184234939541668608015960 91,\ 0.03879536961277926055922378638407792557 768]$$

$$choose\_vector\_dS := [0.01576111812428889709138079692984191481 274,\ 0.96120463038722073944077621361592207442 18] \quad (5.8.3)$$

```
> choose_vector_dP:=[evalf(y0*(dSeval+dHeval)*exp(Seval+Heval)
/dPeval)];
```

$$choose\_vector\_dP := [0.97629024683284579965958402014004580892 80] \quad (5.8.4)$$

```
> choose_vector_dH:=[evalf(dxKeval/dHeval)];
```

$$choose\_vector\_dH := [0.90731473996739180028952510113841356351 70] \quad (5.8.5)$$

```
> ch_dP_or_dH:=[evalf(dPeval/(dPeval+dHeval))];
```

$$ch\_dP\_or\_dH := [0.98153083276505528051601966894825117251 20] \quad (5.8.6)$$

```
> ch_dS_or_dH:=[evalf(dSeval/(dSeval+dHeval))];
```

$$ch\_dS\_or\_dH := [0.98799858617920087914129630299712716197 50] \quad (5.8.7)$$

Calculating the choose-vectors of bi-derivate networks

We recall the system verified by bi-pointed networks

```
> equations_bi_derivate_network:={
ddD=ddS+ddP+ddH,
ddS=(ddP+ddH)*x*D+2*(dP+dH)*D+2*(dP+dH)*x*dD+2*(y+P+H)*dD+(y+
P+H)*x*ddD,
ddP=y*(ddS+ddH)*exp(S+H)+y*(dS+dH)^2*exp(S+H)+(ddS+ddH)*(exp
(S+H)-1)+(dS+dH)^2*exp(S+H),
ddH=dxxK+2*dD*dxyK+ddD*dyK+dD^2*dyyK
}:
> solution_bi_derivate_network:=eval_bi_derivate_network(x0,
y0);solutions_bi_derivate_3_connected:=subs({x=x0,op
(eval_network(x0,y0))},system_bi_derivate_3_connected);
```

$$solution\_bi\_derivate\_network := \{ddD = 2.81920848804660839677743936033568320496 1,\ ddH = 0.20416974306635920499362807814272606204 02,\ ddP = 1.27876199775176316272427578102855399812 2,\ ddS = 1.33627674722848602905953550116440314479 9\}$$

$$solutions\_bi\_derivate\_3\_connected := \{dxxK = 0.14525691617753654702381747274670685673 47,\ dxyK = 0.03928066227449508126801960756239290835 536,\ dyyK = 0.00468811527622347696103753301794425385 6877\} \quad (5.9.1)$$

```
> ddDeval:=subs(solution_bi_derive_network,ddD):ddSeval:=
  subs(solution_bi_derive_network,ddS):ddPeval:=subs
  (solution_bi_derive_network,ddP):ddHeval:=subs
  (solution_bi_derive_network,ddH):dxxKeval:=subs
  (solutions_bi_derive_3_connected,dxxK):dxyKeval:=subs
  (solutions_bi_derive_3_connected,dxyK):dyyKeval:=subs
  (solutions_bi_derive_3_connected,dyyK):
> choose_vector_ddD_Bernoulli:=[evalf(ddSeval/ddDeval),evalf
  (ddPeval/ddDeval),evalf(ddHeval/ddDeval)];
  choose_vector_ddD:=[choose_vector_ddD_Bernoulli[1],
  choose_vector_ddD_Bernoulli[1]+choose_vector_ddD_Bernoulli[2]
  ];
```

$$choose\_vector\_ddD\_Bernoulli := [0.47399004113894896628644676609365583498 50,\ 0.45358901378656111154409390687461254437 26,\ 0.07242094507448992216945932703173162064246]$$

$$choose\_vector\_ddD := [0.47399004113894896628644676609365583498 50,\ 0.92757905492551007783054067296826837935 76] \quad \textbf{(5.9.2)}$$

```
> choose_vector_ddS_Bernoulli:=[evalf((ddPeval+ddHeval)*x0*
  Deval/ddSeval),evalf(2*(dPeval+dHeval)*Deval/ddSeval),evalf
  (2*(dPeval+dHeval)*x0*dDeval/ddSeval),evalf(2*(y0+Peval+
  Heval)*dDeval/ddSeval),evalf((y0+Peval+Heval)*x0*
  ddDeval/ddSeval)];
  choose_vector_ddS:=[choose_vector_ddS_Bernoulli[1],
  choose_vector_ddS_Bernoulli[1]+choose_vector_ddS_Bernoulli
  [2],choose_vector_ddS_Bernoulli[1]+
  choose_vector_ddS_Bernoulli[2]+choose_vector_ddS_Bernoulli
  [3],choose_vector_ddS_Bernoulli[1]+
  choose_vector_ddS_Bernoulli[2]+choose_vector_ddS_Bernoulli[3]
  +choose_vector_ddS_Bernoulli[4]];
```

$$choose\_vector\_ddS\_Bernoulli := [0.02659395929162126041222726643445977991876,\ 0.26382092598250862937085106373746481682 04,\ 0.01082563918660715978322068373327161297560,\ 0.64938478693997359889362732429350130366 30,\ 0.04937468859928935154007366180130248662301]$$

$$choose\_vector\_ddS := [0.02659395929162126041222726643445977991876,\ 0.29041488527412988978307833017192459673 92,\ 0.30124052446073704956629901390519620971 48,\ 0.95062531140071064845992633819869751337 78] \quad \textbf{(5.9.3)}$$

```
> choose_vector_ddP_Bernoulli:=[evalf(y0*(ddSeval+ddHeval)*exp
  (Seval+Heval)/ddPeval),evalf(y0*(dSeval+dHeval)^2*exp(Seval+
  Heval)/ddPeval),evalf((ddSeval+ddHeval)*(exp(Seval+Heval)-1)
  /ddPeval),evalf((dSeval+dHeval)^2*exp(Seval+Heval)/ddPeval)];

  choose_vector_ddP:=[choose_vector_ddP_Bernoulli[1],
  choose_vector_ddP_Bernoulli[1]+choose_vector_ddP_Bernoulli
  [2],choose_vector_ddP_Bernoulli[1]+
  choose_vector_ddP_Bernoulli[2]+choose_vector_ddP_Bernoulli[3]
  ];
```

$$choose\_vector\_ddP\_Bernoulli := [0.75010988747681722162475310902907119013 93,\ 0.08808081066364047881941273649474816041168,\ 0.01821683698880807192893311071492020939812,\ 0.14359246487073422762690104376126044004 87]$$

$$choose\_vector\_ddP := [0.75010988747681722162475310902907119013 93,\ 0.83819069814045770044416584552381935055 10, \quad \textbf{(5.9.4)}$$

0.85640753512926577237309895623873955994911]

```
> choose_vector_ddH_Bernoulli:=[dxxKeval/ddHeval,2*dDeval*
  dxyKeval/ddHeval,ddDeval*dyKeval/ddHeval,dDeval^2*
  dyyKeval/ddHeval];
  choose_vector_ddH:=[choose_vector_ddH_Bernoulli[1],
  choose_vector_ddH_Bernoulli[1]+choose_vector_ddH_Bernoulli
  [2],choose_vector_ddH_Bernoulli[1]+
  choose_vector_ddH_Bernoulli[2]+choose_vector_ddH_Bernoulli[3]
  ];
```

$$choose\_vector\_ddH\_Bernoulli := [0.711451726372233206319958280529255303638 6,\ 0.268028860660535740927967988412636145990 6,\ 0.009378130358228438686051949251578737774831,\ 0.011141282609002614066021781806529812595 24]$$

$$choose\_vector\_ddH := [0.711451726372233206319958280529255303638 6,\ 0.979480587032768947247926268941891449629 2,\ 0.988858717390997385933978218193470187404 0] \quad (5.9.5)$$

```
> ch_ddP_or_ddH:=[evalf(ddPeval/(ddPeval+ddHeval))];
```

$$ch\_ddP\_or\_ddH := [0.862320201634014332738252474670475419827 6] \quad (5.9.6)$$

```
> ch_ddS_or_ddH:=[evalf(ddSeval/(ddSeval+ddHeval))];
```

$$ch\_ddS\_or\_ddH := [0.867460671725584823314889013723662244650 3] \quad (5.9.7)$$

```
> solution_planar_graphs:=evaluate_planar_and_connected(z0,x0,y0)
  ;
```

$$solution\_planar\_graphs := \{C = 0.037135097411061243157710617643762547900 37,\ G = 1.037833219948803835342298837456808254150,\ dC = 1.023491521464250900413721480849623130282,\ dG = 1.062213501311543786405627023898466251235,\ ddC = 0.668125257668682662045131995384991297 85,\ ddG = 1.780569100072634188826786481790089062648,\ dddC = 1.637076101793299527135976557021690987,\ dddG = 4.940792677856805751516501823866443370869\} \quad (5.2)$$

Calculating the choose-vectors of 2-connected planar graphs

```
> B_eval:=eval_2connected(x0,y0);dB_eval:=
  eval_derive_2connected(x0,y0);ddB_eval:=
  eval_bi_derive_2connected(x0,y0);dddB_eval:=
  eval_tri_derive_2connected(x0,y0);
```

$$B\_eval := 0.000435099886179432767849590670288441066 46$$

$$dB\_eval := 0.023219842219972369437856348855081731950 7$$

$$ddB\_eval := 0.622880478267134011353588999105453800862 9$$

$$dddB\_eval := 0.276424314333324136989948497000437113980 0 \quad (5.10.1)$$

```
> ch_xy_in_dB:=[x0*y0/dB_eval];
```

$$ch\_xy\_in\_dB := [0.992566456633208887302945378903327480603 3] \quad (5.10.2)$$

```
> ch_y_in_ddB:=[y0/ddB_eval];
```

$$ch\_y\_in\_ddB := [0.984792954076464890371157619914397978558 4] \quad (5.10.3)$$

```
> ch_nontrivialD_or_dD:=[(D_eval-y0)/(x0*dDeval+D_eval-y0)];
```

$$ch\_nontrivialD\_or\_dD := [0.482465532258265191015714847536343779327 8] \quad (5.10.4)$$

```
> ch_dD_or_ddD:=[dDeval/(dDeval+x0*ddDeval)];
```

$$ch\_dD\_or\_ddD := [0.868005791030429935585449433059733104334 9] \quad (5.10.5)$$

Calculating the choose-vectors of connected planar graphs

```
> C_eval:=subs(solution_planar_graphs,C);dC_eval:=subs
  (solution_planar_graphs,dC);ddC_eval:=subs
```

```
(solution_planar_graphs,ddC);dddC_eval:=subs
(solution_planar_graphs,dddC);
```

$$C\_eval := 0.037135097411061243157710617643762547900 37$$

$$dC\_eval := 1.023491521464250900413721480849623130282$$

$$ddC\_eval := 0.668125257668682662045131995384991297 85$$

$$dddC\_eval := 1.637076101793299527135976557021690987 \quad (5.11.1)$$

```
> exp_dB:=evalf(exp(dB_eval)):
poisson_dB:=[seq(0,i=1..22)]:
poisson_dB[1]:=evalf(1/exp_dB):
for i from 2 to 22 do poisson_dB[i]:=poisson_dB[i-1]+evalf
(dB_eval^(i-1)/(i-1)!/exp_dB):
od:
poisson_dB;
```

$$[0.97704766383346007447597907747456152573 96, \quad (5.11.2)$$
$$0.99973455642906562138578790124173027165 60,$$
$$0.99999794946233133149229133136180527785 62,$$
$$0.99999998811055608806639522975857213566 09,$$
$$0.99999999994482861828500013070744194319 57,$$
$$0.99999999999978660647296597076863801847 25,$$
$$0.99999999999999929244204124949433448686 86,$$
$$0.99999999999999999794699043941820085054 36,$$
$$0.99999999999999999999470464011833450751 22,$$
$$0.99999999999999999999998770685742938946 26,$$
$$0.99999999999999999999999997405504132617 16,$$
$$0.99999999999999999999999999949804328934 5,$$
$$0.99999999999999999999999999999910354871 8,$$
$$0.99999999999999999999999999999998513346,$$
$$0.99999999999999999999999999999999997699,$$
$$0.99999999999999999999999999999999999997,$$
$$1.00000000000000000000000000000000000000,$$
$$1.00000000000000000000000000000000000000,$$
$$1.00000000000000000000000000000000000000,$$
$$1.00000000000000000000000000000000000000,$$
$$1.00000000000000000000000000000000000000,$$
$$1.00000000000000000000000000000000000000]$$

```
> ch_dC_or_ddC:=[evalf(dC_eval/(z0*ddC_eval+dC_eval))];
```

$$ch\_dC\_or\_ddC := [0.97659687102709445267308527043216909971 93] \quad (5.11.3)$$

```
> ch_2ddC_or_dddC:=[evalf(2*ddC_eval/(z0*dddC_eval+2*ddC_eval)
)];
```

$$ch\_2ddC\_or\_dddC := [0.95696118904405137025363617990836649912 80] \quad (5.11.4)$$

```
> choose_vector_dddC_Bernoulli:=[evalf((2*ddC_eval+z0*
dddC_eval)*ddB_eval*dC_eval/dddC_eval),evalf((dC_eval+z0*
ddC_eval)^2*dddB_eval*dC_eval/dddC_eval),evalf((dC_eval+z0*
ddC_eval)*ddB_eval*ddC_eval/dddC_eval)];
choose_vector_dddC[1]+choose_vector_dddC[2]+
choose_vector_dddC[3];
choose_vector_dddC:=[choose_vector_dddC_Bernoulli[1],
choose_vector_dddC_Bernoulli[1]+choose_vector_dddC_Bernoulli
[2]];
```

$$choose\_vector\_dddC\_Bernoulli :=$$
$$[0.54376801898312835973156376906136907269 02,$$
$$0.18981453414191337379177991394738512648 40,$$
$$0.26641744687495826647665631699124580065 11]$$

$$choose\_vector\_dddC_1 + choose\_vector\_dddC_2 + choose\_vector\_dddC_3$$

$$choose\_vector\_dddC := [0.543768018983128359731563769061369072690 2,\ 0.733582553125041733523343683008754199174 2] \quad (5.11.5)$$

Calculating the choose-vectors of planar graphs

```
> G_eval:=subs(solution_planar_graphs,G);dG_eval:=subs
  (solution_planar_graphs,dG);ddG_eval:=subs
  (solution_planar_graphs,ddG);dddG_eval:=subs
  (solution_planar_graphs,dddG);
```

$$G\_eval := 1.037833219948803835342298837456808254150$$
$$dG\_eval := 1.062213501311543786405627023898466251235$$
$$ddG\_eval := 1.780569100072634188826786481790089062648$$
$$dddG\_eval := 4.940792677856805751516501823866443370869 \quad (5.12.1)$$

```
> exp_C:=evalf(exp(C_eval)):
  poisson_C:=[seq(0,i=1..21)]:
  poisson_C[1]:=evalf(1/exp_C):
  for i from 2 to 21 do
  poisson_C[i]:=poisson_C[i-1]+evalf(C_eval^(i-1)/(i-1)!/exp_C)

  od:
  poisson_C;
```

$$[0.963545953991846452280481366493544942783 7,\ 0.999327326853367605182493555316732654576 3,\ 0.999991699236724650777950809578711576280 7,\ 0.999999230811157117151651849622360039508,\ 0.999999994294313505795490079157810035355,\ 0.999999999964717782634848884396776951779,\ 0.999999999999981295183160353004590012631 5,\ 0.999999999999999913219257198143593283689 6,\ 0.999999999999999999642080205518430032345 5,\ 0.999999999999999999998671311175959959082 3,\ 0.999999999999999999999955157204109996623,\ 0.999999999999999999999999861262973959687,\ 0.999999999999999999999999999603772341611,\ 0.999999999999999999999999999998949189598,\ 0.999999999999999999999999999999997398937,\ 0.999999999999999999999999999999999993961,\ 0.999999999999999999999999999999999999984,\ 0.999999999999999999999999999999999999997,\ 0.999999999999999999999999999999999999997,\ 0.999999999999999999999999999999999999997,\ 0.999999999999999999999999999999999999997] \quad (5.12.2)$$

```
> choose_vector_ddG:=[evalf(ddC_eval*G_eval/ddG_eval)];
```

$$choose\_vector\_ddG := [0.389427508018154093796843222228122747817 6] \quad (5.12.3)$$

```
> choose_vector_dddG_Bernoulli:=[evalf(dddC_eval*
  G_eval/dddG_eval),evalf(2*ddC_eval*dG_eval/dddG_eval),evalf
  (dC_eval*ddG_eval/dddG_eval)];
  choose_vector_dddG:=[choose_vector_dddG_Bernoulli[1],
  choose_vector_dddG_Bernoulli[1]+choose_vector_dddG_Bernoulli
  [2]];
```

$$choose\_vector\_dddG\_Bernoulli :=$$
$$[0.343874368507679495225038319962702184155 6,$$

0.28727846543473083290462466607501534380 68,
0.36884716605758967187033701396228247203 75]

$choose\_vector\_dddG := [0.34387436850767949522503831996270218415 56,$ **(5.12.4)**
0.63115283394241032812966298603771752796 24]

All choose-vectors

```
> ch_1_or_u;
  ch_1_or_v;
  ch_u_or_v;
  ch_dxu_or_dxv;
  choose_vector_dxu;
  choose_vector_dxv;
  ch_dyu_or_dyv;
  choose_vector_dyv;
  choose_vector_dyu;
  ch_K_in_dyK;
  ch_dxK_in_dxyK;
  ch_b_or_dxb;
  ch_3b_or_dyb;
  choose_vector_non_trivial_D;
  choose_vector_D;
  choose_vector_P;
  ch_y_or_P_or_H;
  ch_S_or_H;
  poisson_S_plus_H;
  poisson_at_least_1_S_plus_H;
  poisson_at_least_2_S_plus_H;
  choose_vector_dD;
  choose_vector_dS;
  choose_vector_dP;
  choose_vector_dH;
  ch_dP_or_dH;
  ch_dS_or_dH;
  choose_vector_ddD;
  choose_vector_ddS;
  choose_vector_ddP;
  choose_vector_ddH;
  ch_ddP_or_ddH;
  ch_ddS_or_ddH;
  ch_xy_in_dB;
  ch_y_in_ddB;
  ch_nontrivialD_or_dD;
  ch_dD_or_ddD;
  poisson_dB;
  ch_dC_or_ddC;
  ch_2ddC_or_dddC;
  choose_vector_dddC;
  poisson_C;
  choose_vector_ddG;
  choose_vector_dddG;
```

[0.93286813210843925863755845937409772383 21]
[0.57706520408645185403226336431748059815 25]
[0.08940938633862086548106554134360917655 423]
[0.42240252403975052044002916518615778799 79]

[0.8864303886159499286965656876358170718679,
0.9432151943079749643482828438179085359344]
[0.5]
[0.13777420794161507226803628047524164369 77]
[0.7815028909514930634548144361419184856587,
0.8907514454757465317274072180709592428296]
[0.4802237344072258778919033682803336032049,
0.7401118672036129389459516841401668016025]
[0.1977324220326980778121815773664505247694]
[0.1922044499705513539896246196855002728830]
[0.8072408135148624632138930717142420254920]
[0.6894393716754773376399207862913805338496]
[0.6117898469830750784412159443584047656820,
0.9966236487047868229739017229788379518701]
[0.9617464873014680939111760153116325546148,
0.9851495979818680481601054070193035293086,
0.9998708427028538580842032950466872473185]
[0.9879436017149357694658587183321407335145]
[0.9847937238173282515174904608537235313935,
0.9998677475849466134866061886007017365434]
[0.9945114811173584887361633642087443850972]
[0.9851030371566665915285064911050067612741,
0.9998884851201868812323145735288634828982,
0.9999994427881852185284533969143764750303,
0.9999999791076781935376046952710643524 08,
0.9999999999937316690312873890360030770872,
0.9999999999998432534187859503640128517 63,
0.9999999999999999664002293471054943864563,
0.9999999999999999993697563029086392855 00,
0.9999999999999999999998949136466212829099,
0.9999999999999999999999998422970454685272,
0.9999999999999999999999999997848454459906,
0.9999999999999999999999999999997309206915,
0.9999999999999999999999999999999996893621,
0.9999999999999999999999999999999999996666,
0.9999999999999999999999999999999999999993,
0.9999999999999999999999999999999999999996,
0.9999999999999999999999999999999999999996,
0.9999999999999999999999999999999999999996,
0.9999999999999999999999999999999999999996,
0.9999999999999999999999999999999999999996,
0.9999999999999999999999999999999999999996]
[0.9925142540136613700112366830991221847445,
0.9999625956095452818197007547143752856029,
0.9999998597544880377278330641985219953995,
0.9999999957922087645417109270699885587 00,
0.9999999999989477950447852808977501194517,
0.9999999999999977445220877401291528260160,
0.9999999999999999957693141634343052813727,
0.9999999999999999999929457867027073475742,
0.9999999999999999999999894137512330543891,
0.9999999999999999999999999855571530739365,

0.99999999999999999999999999819373041724,
0.99999999999999999999999999791475614,
0.99999999999999999999999999999776149,
0.99999999999999999999999999999999463,
0.99999999999999999999999999999999686,
0.99999999999999999999999999999999686,
0.99999999999999999999999999999999686,
0.99999999999999999999999999999999686,
0.99999999999999999999999999999999686,
0.99999999999999999999999999999999686,
0.99999999999999999999999999999999686]
[0.995003251443078537371096704643804702489 5,
0.99998126499186344045800473710201730468 25,
0.99999943789286423324507128666882717710 5,
0.99999999859438864592122577728868196659 1,
0.99999999999698696974706851800580912613 6,
0.99999999999999434834438105884092618100 4,
0.99999999999999999057647252502758984928 6,
0.99999999999999999998585812451255319182 9,
0.99999999999999999999998070620222437332 0,
0.99999999999999999999999975870546756835,
0.99999999999999999999999999972143874808,
0.99999999999999999999999999999970164319,
0.99999999999999999999999999999999996163,
1.00000000000000000000000000000000002601,
1.00000000000000000000000000000000002604,
1.00000000000000000000000000000000002604,
1.00000000000000000000000000000000002604,
1.00000000000000000000000000000000002604,
1.00000000000000000000000000000000002604,
1.00000000000000000000000000000000002604,
1.00000000000000000000000000000000002604]
[0.603244946858063439921322275877838485164 8,
0.992672264572212194304217373784407659668 2]
[0.015761118124288897091380796929841914812 74,
0.961204630387220739440776213615922074421 8]
[0.976290246832845799659584020140045808928 0]
[0.907314739967391800289525101138413563517 0]
[0.981530832765055280516019668948251172512 0]
[0.987998586179200879141296302997127161975 0]
[0.473990041138948966286446766093655834985 0,
0.927579054925510077830540672968268379357 6]
[0.026593959291621260412227266434459779918 76,
0.290414885274129889783078330171924596739 2,
0.301240524460737049566299013905196209714 8,
0.950625311400710648459926338198697513377 8]
[0.750109887476817221624753109029071190139 3,
0.838190698140457700444165845523819350551 0,
0.856407535129265772373098956238739559949 1]
[0.711451726372233206319958280529255303638 6,
0.979480587032768947247926268941891449629 2,
0.988858717390997385933978218193470187404 0]

[0.8623202016340143327382524746704754198276]
[0.8674606717255848233148890137236622446503]
[0.9925664566332088730294537890332748060 33]
[0.9847929540764648903711576199143979785584]
[0.4824655322582651910157148475363437793278]
[0.8680057910304299355854494330597331043349]
[0.9770476638334600744759790774745615257396,
0.9997345564290656213857879012417302716560,
0.9999979494623313314922913313618052778562,
0.9999999881105560880663952297585721356609,
0.9999999999448286182850001307074419431957,
0.9999999999997866064729659707686380184725,
0.9999999999999992924420412494334484868686,
0.9999999999999999979469904394182008505436,
0.9999999999999999999470464011833450751 22,
0.9999999999999999999998770685742938946 26,
0.9999999999999999999999997405504132617 16,
0.9999999999999999999999999498043289345,
0.9999999999999999999999999999103548718,
0.9999999999999999999999999999998513346,
0.9999999999999999999999999999999997699,
0.9999999999999999999999999999999999997,
1.0000000000000000000000000000000000000,
1.0000000000000000000000000000000000000,
1.0000000000000000000000000000000000000,
1.0000000000000000000000000000000000000,
1.0000000000000000000000000000000000000,
1.0000000000000000000000000000000000000]
[0.9765968710270944526730852704321690997193]
[0.9569611890440513702536361799083664991280]
[0.5437680189831283597315637690613690726902,
0.7335825531250417335233436830087541991742]
[0.9635459539918464522804813664935449427837,
0.9993273268533676051824935553167326545763,
0.9999916992367246507779508095787115762807,
0.9999999230811157117151651849622360039508,
0.9999999994294313505795490079157810035355,
0.9999999999964717782634848884396776951779,
0.9999999999999812951831603530045900126315,
0.9999999999999999132192571981435932836896,
0.9999999999999999996420802055184300323455,
0.9999999999999999999867131111759599590823,
0.9999999999999999999999955157204109996623,
0.9999999999999999999999999861262973959687,
0.9999999999999999999999999999603772341611,
0.9999999999999999999999999999998949189598,
0.9999999999999999999999999999999997398937,
0.9999999999999999999999999999999999993961,
0.9999999999999999999999999999999999999984,
0.9999999999999999999999999999999999999997,
0.9999999999999999999999999999999999999997,
0.9999999999999999999999999999999999999997,

0.99999999999999999999999999999999999997]
[0.38942750801815409379684322222812274781 76]
[0.34387436850767949522503831996270218415 56,
0.63115283394241032812966298603771752796 24]   (5.13.1)