```
> 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

```
> solve(system_u_v, {u,v});
```

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};
```

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

```
> 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)
  };
```

$$system\_derivate\_binary\_tree := \left\{ dxu = -\frac{\mathrm{D}\left(v^2+2v+1\right)}{4\,\mathrm{D}^2 u v x + 4\,\mathrm{D}^2 u x + 4\,\mathrm{D}^2 v x + 4\,\mathrm{D}^2 x - 1},\ dxv = -\frac{2\,\mathrm{D}^2 (1+u)\left(v^2+2v+1\right)}{4\,\mathrm{D}^2 u v x + 4\,\mathrm{D}^2 u x + 4\,\mathrm{D}^2 v x + 4\,\mathrm{D}^2 x - 1},\ dyu = -\frac{x\left(2\,\mathrm{D}\,u^2 v + 2\,\mathrm{D}\,u^2 + 4\,\mathrm{D}\,u v + 4\,\mathrm{D}\,u + 2\,\mathrm{D}\,v + v^2 + 2\,\mathrm{D} + 2v + 1\right)}{4\,\mathrm{D}^2 u v x + 4\,\mathrm{D}^2 u x + 4\,\mathrm{D}^2 v x + 4\,\mathrm{D}^2 x - 1},\ dyv = \right. \quad \textbf{(1.2.2)}$$

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

```
> system_bi_derive_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_derivate_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))};
```

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

```
> 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)))
};
```

$$system\_derivate\_3\_connected := \left\{ dxK = \frac{1}{2}\left(\mathrm{D}\left(-\frac{\mathrm{D}}{(\mathrm{D}\,x+1)^2} \right.\right.\right. \qquad \textbf{(1.3.2)}$$

$$+ \frac{2\,(1+u)\,(1+v)^2\,\mathrm{D}\,(v^2+2\,v+1)}{(1+u+v)^3\,(4\,\mathrm{D}^2\,u\,v\,x+4\,\mathrm{D}^2\,u\,x+4\,\mathrm{D}^2\,x\,v+4\,\mathrm{D}^2\,x-1)}$$

$$+ \frac{4\,(1+u)^3\,(1+v)\,\mathrm{D}^2\,(v^2+2\,v+1)}{(1+u+v)^3\,(4\,\mathrm{D}^2\,u\,v\,x+4\,\mathrm{D}^2\,u\,x+4\,\mathrm{D}^2\,x\,v+4\,\mathrm{D}^2\,x-1)}$$

$$+ \frac{1}{(1+u+v)^4}\left(3\,(1+u)^2\,(1+v)^2\left(\right.\right.$$

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

$$-\frac{2\,\mathrm{D}^2(1+u)\left(v^2+2v+1\right)}{4\,\mathrm{D}^2 u v x+4\,\mathrm{D}^2 u x+4\,\mathrm{D}^2 x v+4\,\mathrm{D}^2 x-1}\Bigg)\Bigg)\Bigg)\Bigg),\ dyK=\frac{1}{2\,(\mathrm{D}x+1)}$$

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

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

$$\left.\left.+2\,\mathrm{D}v+v^2+2\,\mathrm{D}+2v+1\right)\right)\Big/\left((1+u+v)^3\left(4\,\mathrm{D}^2 u v x+4\,\mathrm{D}^2 u x+4\,\mathrm{D}^2 x v\right.\right.$$

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

$$\left.\left.+2\,\mathrm{D}u x+4\,\mathrm{D}v x+2\,\mathrm{D}x+u^2+2u+1\right)\right)\Big/\left((1+u+v)^3\left(4\,\mathrm{D}^2 u v x\right.\right.$$

$$\left.\left.+4\,\mathrm{D}^2 u x+4\,\mathrm{D}^2 x v+4\,\mathrm{D}^2 x-1\right)\right)+\frac{1}{(1+u+v)^4}\left(3\,(1+u)^2(1+v)^2\left(\right.\right.$$

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

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

$$\left.\left.\left.\right)\right)\right)\Bigg\}$$

```
> 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);
```

$$\{t = 0.62637166330645166589299785045039561167 21, xsi = 0.03819109766941133539115256404542235955388\} \quad \textbf{(1.5.1.1)}$$

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);
```

$$\{D\_0 = 1.09417463309857950144276906088188635683 6, D\_2 = -0.13749382418806273997533867902060211970 97, D\_3 = 0.05432940491186193454144415400902722147575\} \quad \textbf{(1.5.2.1)}$$

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);
```

$approx = 2.10283909576108252327553250123626094166 1$, $exact$ **(1.5.3.1.1)**
$= 2.10283909576108250374483013473725881157 0$

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_derivate_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_derive_network(Rk,Y0),dddD):
approx=compare[1],exact=compare[2];
end proc:
```

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

$$approx = 1.025918116770318952601982006021201761795\ 10^{24},\ exact = 1.025918116770027417088155919876806895586\ 10^{24} \quad \textbf{(1.5.3.4.1)}$$

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;
```

$$y0 := 1 \quad \textbf{(1.6.1.1)}$$

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))
  );
```

$$R := 0.038191097669411335391152564045422359553 88$$
$$x0 := 0.038191078573862500685484868469140336842 70 \quad \textbf{(1.6.1.2)}$$

Calculating the choose-vectors of bicolored binary trees

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

$$solutions\_networks := \{D = 1.094174564370870916793206753393405522040,\ H = 0.00212326831809155637201619222985965062051 8,\ K = 0.00212326831809155637201619222985965062051 8,\ P = 0.04816227188310305699128946641715724277490,\ S = 0.04388902416967630342990109474638862864503,\ u = 0.53150494740292793874623740445440171690 75,\ v = 2.56639454191243001904884671516936829135 1\} \quad \textbf{(1.6.2.1)}$$

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

$$ch\_1\_or\_u := [0.65295251033681916726502365056303868289 34] \quad \textbf{(1.6.2.2)}$$

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

(1.6.2.3)

$$ch\_1\_or\_v := [0.28039522499486659558475591879220540105 92]$$ (1.6.2.3)

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

$$ch\_u\_or\_v := [0.17156946157746118654760754670349873172 19]$$ (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);
```

$$solution\_derivate\_binary\_tree := \{dxu = 13230.625331547239353911536224560914356 13, dxv = 44342.011032417321120430334811027485858 96, dyu = 1126.431512981459152507965343199629438780, dyv = 3777.544133638864258208363721570382908734\}$$ (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]];
```

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

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

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

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

```
> 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))];
```

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

Calculating the choose-vectors of 3-connected networks

```
> solution_derive_3_connected:=subs({x=x0,op(eval_network
(x0,y0))},system_derive_3_connected);
solution_bi_derive_3_connected:=subs({x=x0,op
(eval_network(x0,y0))},system_bi_derive_3_connected);
```

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

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

```
> K_eval:=subs(solutions_networks,K);dxK_eval:=subs
(solution_derive_3_connected,dxK);dyK_eval:=subs
(solution_derive_3_connected,dyK);dxyK_eval:=subs
(solution_bi_derive_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);
```

$$solutions\_networks := \{D = 1.09417456437087091679320675339340552204 0, H = 0.00212326831809155637201619222985965062051 8, K = 0.00212326831809155637201619222985965062051 8, P = 0.04816227188310305699128946641715724277490, S$$ (1.6.5.1)

$= 0.043889024169676303429901094746388628645 03, u$
$= 0.531504947402927938746237404454401716907 5, v$
$= 2.566394541912430019048846715169368291351\}$

```
> 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]];
```

$$choose\_vector\_non\_trivial\_D\_Bernoulli := [0.46603904634839589009660545863589332516 95, 0.51141486244028863352516178359168337438 9, 0.02254609121131547637823266300493833739631]$$

$$choose\_vector\_non\_trivial\_D := [0.46603904634839589009660545863589332516 95, 0.97745390878868452362176733699506166260 84] \quad (1.6.5.2)$$

```
> 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]];
```

$$choose\_vector\_D\_Bernoulli := [0.91393095083962268965377078684537679053 53, 0.04011153759081544570580073793828548527298, 0.04401699093672080208228049252804957907682, 0.00194052063284106255814798268828814511534 4]$$

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

$$choose\_vector\_D := [0.91393095083962268965377078684537679053 53, 0.95404248843043813535957152478366227580 83, 0.99805947936715893744185201731171185488 51] \quad (1.6.5.3)$$

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

$$choose\_vector\_P := [0.97767983827098610986898981012310840309 50] \quad (1.6.5.4)$$

```
> 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]];
```

$$ch\_y\_or\_P\_or\_H\_Bernoulli := [0.95212202941348519033925565531451712737 71, 0.04585636004650411959540501051019333820209, 0.00202161054001069006533933417528953442029 3]$$

$$ch\_y\_or\_P\_or\_H := [0.95212202941348519033925565531451712737 71, 0.99797838945998930993466066582471046557 92] \quad (1.6.5.5)$$

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

$$ch\_S\_or\_H := [0.95385432450130545561815998820163951601 76] \quad (1.6.5.6)$$

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;
```

[0.955030222421232247306303084518528824 0903, **(1.6.5.7)**
0.998973352349935980105730409838607548 3163,
0.999984314423489730992419722564588585 5676,
0.999999819984363862815903665606290938 7042,
0.999999998345964394676641475423760922 3353,
0.999999999873296211283190221659581348 341,
0.999999999999916783941443341304319751 5223,
0.999999999999999521686721717171581381 9236,
0.999999999999999997555888658054660065 0439,
0.999999999999999999988758801965142738 2331,
0.999999999999999999999529952297794396 222,
0.999999999999999999999998198202028091 558,
0.999999999999999999999999993624319908 235,
0.999999999999999999999999999979050341 322,
0.999999999999999999999999999999935749 563,
0.999999999999999999999999999999999815 264,
0.999999999999999999999999999999999999 502,
1.000000000000000000000000000000000000 000,
1.000000000000000000000000000000000000 000,
1.000000000000000000000000000000000000 000,
1.000000000000000000000000000000000000 000]

```
> 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;
```

[0.977170275119422732949676243281663325 1317, **(1.6.5.8)**
0.999651197373996466658161271439529760 4411,
0.999995996964053872093298531181184285 8165,
0.999999963218950718041735231617012804 3908,
0.999999999718246796985199421582304403 9487,
0.999999999998149511448863631332802981 4647,
0.999999999999989363672581101189328386 8903,
0.999999999999999945649912595980585151 0973,
0.999999999999999999750027715499203733 4195,
0.999999999999999999998954747549324027 8726,
0.999999999999999999999995993313578764 1720,

0.99999999999999999999999958223001423642,
0.99999999999999999999999995341391528886,
0.99999999999999999999999999998571252956,
0.99999999999999999999999999999995891998,
0.99999999999999999999999999999999988930,
1.00000000000000000000000000000000000002,
1.00000000000000000000000000000000000005,
1.00000000000000000000000000000000000005,
1.00000000000000000000000000000000000005,
1.00000000000000000000000000000000000005]

```
> 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;
```

[0.98472155806396587637157844913103405 28525, **(1.6.5.9)**
0.99982465684684910285801679211064320 29449,
0.99999838889651652130519179918773821 38919,
0.99999998765849328063929637520627804 16867,
0.99999999991894389613469675552183181 34076,
0.99999999999953410181355501893372913 83826,
0.99999999999999761932797314353219090 73569,
0.99999999999999998905057832241053593 38723,
0.99999999999999999995421528484711454 34885,
0.99999999999999999999824496946757139 0968,
0.99999999999999999999999378980695922 0950,
0.99999999999999999999999999795941103 34049,
0.99999999999999999999999999999374172 50766,
0.99999999999999999999999999999998200 63301,
0.99999999999999999999999999999999995 19330,
1.00000000000000000000000000000000000505,
1.00000000000000000000000000000000000629,
1.00000000000000000000000000000000000629,
1.00000000000000000000000000000000000629,
1.00000000000000000000000000000000000629,
1.00000000000000000000000000000000000629]

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* **(1.6.6.1)**

$= 3.59864541423366329023834517989935262332 8, dH$
$= 0.33240343578292901802664306370630343525 95, dP$
$= 1.88023784904830183671547258423502766503 8, dS$
$= 1.38600412940243243549622953195802152303 1\}$

```
> 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]];
```

$choose\_vector\_dD\_Bernoulli :=$
$[0.38514606743981860341072268015078050071 97,$
$0.52248488878938387634684845774073558795 51,$
$0.09236904377079752024242886210848391132 536]$

$choose\_vector\_dD := [0.38514606743981860341072268015078050071 97,$ **(1.6.6.2)**
$0.90763095622920247975757113789151608867 48]$

```
> 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]];
```

$choose\_vector\_dS\_Bernoulli :=$
$[0.06671062749183899407897306600073569750 144,$
$0.82914307326785235226019913838632194581 02,$
$0.10414629924030865366082779561294235668 93]$

$choose\_vector\_dS := [0.06671062749183899407897306600073569750 144,$ **(1.6.6.3)**
$0.89585370075969134633917220438705764331 16]$

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

$choose\_vector\_dP := [0.95696547541981134482688539342268839526 66]$ **(1.6.6.4)**

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

$choose\_vector\_dH := [0.76390824301140713199147365818434542501 45]$ **(1.6.6.5)**

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

$ch\_dP\_or\_dH := [0.84977075223999398539477481719096858072 19]$ **(1.6.6.6)**

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

$ch\_dS\_or\_dH := [0.80656309799993619516744284368662378100 75]$ **(1.6.6.7)**

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_derate_network:=eval_bi_derate_network(x0,
  y0);solutions_bi_derate_3_connected:=subs({x=x0,op
  (eval_network(x0,y0))},system_bi_derate_3_connected);
```

$$solution\_bi\_derate\_network := \{ddD = 39481.662538115071655969411352587448086 33, ddH = 15733.833009982466248744792206267999180 63, ddP = 20631.524985007123046509599739381136236 93, ddS = 3116.304543125482360715019406938312668771\}$$

$$solutions\_bi\_derate\_3\_connected := \{dxxK = 8711.471662296009772585087991766681181610, dxyK = 742.359714935541081809467434246948479 65, dyyK = 63.194420314823533720385184257642938170 00\} \quad (1.6.7.1)$$

```
> ddDeval:=subs(solution_bi_derate_network,ddD):ddSeval:=
  subs(solution_bi_derate_network,ddS):ddPeval:=subs
  (solution_bi_derate_network,ddP):ddHeval:=subs
  (solution_bi_derate_network,ddH):dxxKeval:=subs
  (solutions_bi_derate_3_connected,dxxK):dxyKeval:=subs
  (solutions_bi_derate_3_connected,dxyK):dyyKeval:=subs
  (solutions_bi_derate_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]];
```

$$choose\_vector\_ddD\_Bernoulli := [0.07893042852785247889547584165404340150041, 0.52255968109472906205668207130391357235 41, 0.39850989037741845904784208704204302614 55]$$

$$choose\_vector\_ddD := [0.07893042852785247889547584165404340150041, 0.60149010962258154095215791295795697385 45] \quad (1.6.7.2)$$

```
> 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]];
```

$$choose\_vector\_ddS\_Bernoulli := [0.48763684522444347886674797153333248374 44, 0.00155377356765720348806586283259924690824 8, 0.00019516507119951981400634984620563830791 71, 0.00242569696932779121945352969539438493858 8, 0.50818851916737200661172628609246824610 10]$$

$$choose\_vector\_ddS := [0.48763684522444347886674797153333248374 44, 0.48919061879210068235481383436593173065 26, 0.48938578386330020216882018421213736896 05, 0.49181148083262799338827371390753175389 91] \quad (1.6.7.3)$$

```
> 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]];
```

$$choose\_vector\_ddP\_Bernoulli := [0.956678641721771978534859076070487624518 3, 0.000149866272821146013105562104793584595732 1, 0.043021625732585729438929799719925206290 56, 0.000149866272821146013105562104793584595732 1]$$

$$choose\_vector\_ddP := [0.956678641721771978534859076070487624518 3, 0.956828507994593124547964638175281209114 0, 0.999850133727178853986894437895206415404 6] \quad (1.6.7.4)$$

```
> 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]];
```

$$choose\_vector\_ddH\_Bernoulli := [0.553677648464232544867261701585693847140 6, 0.339585323191067925982500699837291958320 0, 0.054722781947897047005463345253020044622 10, 0.052014246396802482144774253323994149917 31]$$

$$choose\_vector\_ddH := [0.553677648464232544867261701585693847140 6, 0.893262971655300470849762401422985805460 6, 0.947985753603197517855225746676005850082 7] \quad (1.6.7.5)$$

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

$$ch\_ddP\_or\_ddH := [0.567340076449948047776282336853673795433 5] \quad (1.6.7.6)$$

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

$$ch\_ddS\_or\_ddH := [0.165319989540961006361742917234416347038 6] \quad (1.6.7.7)$$

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]
[0.00105187713932158383879344925179959133670 0,
    0.50052593856966079191939672462589979566 84]
                                    [0.5]
                [0.22969761559843038913952606839667943602 21]
[0.00062090800820379697680009889039482160311 75,
    0.50031045400410189848840004944519741080 14]
[0.00043123688955594504460993276563503883165 94,
    0.50021561844477797252230496638281751941 61]
                [0.09736382064286582607785119655750788825 036]
              [0.00034205213387951785063590145517007616783 39]
                [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,
    0.99999981998436386281590366562062909387 042,
    0.99999999834596439467664147542376092233 53,
    0.99999999998732962112831902216595813483 41,
    0.99999999999991678394144334130431975152 23,
    0.99999999999999952168672171717158138192 36,
    0.99999999999999999755588865805466006504 39,
    0.99999999999999999998875880196514273823 31,
    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,
0.99999996321895071804173523161701280439 08,
0.99999999971824679698519942158230440394 87,
0.99999999999814951144886363133280298146 47,
0.99999999999998936367258110118932838689 03,
0.99999999999999994564991259598058515109 73,
0.99999999999999999975002771549920373341 95,
0.99999999999999999999895474754932402787 26,
0.99999999999999999999995959331357876417 20,
0.99999999999999999999999998582230014236 42,
0.99999999999999999999999999995341391528 86,
0.99999999999999999999999999999985712529 56,
0.99999999999999999999999999999999958919 98,
0.99999999999999999999999999999999999889 30,
1.00000000000000000000000000000000000002,
1.00000000000000000000000000000000000005,
1.00000000000000000000000000000000000005,
1.00000000000000000000000000000000000005,
1.00000000000000000000000000000000000005]
[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,
0.99999999999999999999999999795941103340 49,
0.99999999999999999999999999999374172507 66,
0.99999999999999999999999999999998200633 01,
0.99999999999999999999999999999999995193 30,
1.00000000000000000000000000000000000505,
1.00000000000000000000000000000000000629,
1.00000000000000000000000000000000000629,
1.00000000000000000000000000000000000629,
1.00000000000000000000000000000000000629,
1.00000000000000000000000000000000000629]

[0.385146067439818603410722680150780500719 7,
0.907630956229202479757571137891516088674 8]
[0.066710627491838994078973066000735697501 44,
0.895853700759691346339172204387057643311 6]
[0.956965475419811344826885393422688395266 6]
[0.763908243011407131991473658184345425014 5]
[0.849770752239993985394774817190968580721 9]
[0.806563097999936195167442843686623781007 5]
[0.078930428527852478895475841654043401500 41,
0.601490109622581540952157912957956973854 5]
[0.487636845224443478866747971533332483744 4,
0.489190618792100682354813834365931730652 6,
0.489385783863300202168820184212137368960 5,
0.491811480832627993388273713907531753899 1]
[0.956678641721771978534859076070487624518 3,
0.956828507994593124547964638175281209114 0,
0.999850133727178853986894437895206415404 6]
[0.553677648464232544867261701585693847140 6,
0.893262971655300470849762401422985805460 6,
0.947985753603197517855225746676005850082 7]
[0.567340076449948047776282336853673795433 5]
[0.165319989540961006361742917234416347038 6] **(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)

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

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

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

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

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

```
> 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);
```

$$B0 = 0.00073969957112330039054951543726406054544 \qquad \textbf{(2.1.2)}$$

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);
```

$$0.039051802824590761935507953561299040192 3 \tag{2.2.1}$$

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

$$B2 = -0.001491431215840526154995018536741381791414 \tag{2.2.2}$$

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_derivate_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_derivate_2connected(.038191097669411,1);
```

$$1.051966755476522130558698978162973999574 \quad \textbf{(2.3.1)}$$

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

$$B4 = 0.000767178284503197446082025194641497055279 2 \quad \textbf{(2.3.2)}$$

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);
```

$$1.257821982073426377416922 05\ 10^{6} \tag{2.4.1}$$

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

$$R := 0.038191097669411335391152564045422359 55388 \tag{2.4.2}$$

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

$$1.257821982073426377416922 05\ 10^{6} \tag{2.4.3}$$

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

$$B5 = -3.501862271974442506367394183613405071277\ 10^{-6} \quad \text{(2.4.4)}$$

```
> eval_tri_derive_2connected(.038191097669,1);dx:=
  0.00000000000000001;evalf((eval_bi_derive_2connected
  (.038191097669+dx,1)-eval_bi_derive_2connected
  (.038191097669,1))/dx);
```

$$35918.358393714713511116409236400000000001$$

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

$$35918.576687812943818239647000000000000000 \quad \text{(2.4.5)}$$

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;
```

$$inverse\_F := \mathbf{proc}(x, y0) \quad \text{(3.1.1)}$$
$$evalf(x * \exp(-eval\_derivate\_2connected(x, y0)))$$
$$\mathbf{end\ proc}$$

```
> 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);
```

$$0.038191097669402429946807661968678134798 50 \quad \text{(3.1.2)}$$

```
> inverse_F(.038191097669402429946807661968678134798 50,1);
```

$$0.036728412581830000000000000000000000506352 \quad \text{(3.1.3)}$$

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;
```

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

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);
```

$$\{C = 0.037439366024685548490805435227680414199 53,\ G = 1.038149048069666547989910110165655146797,\ dC = 1.039824348093280741233629384704015601806,\ dG = 1.079492657132700989555944229432597122144,\ ddC = 1.185032516943684908500357051427313428 49,\ ddG = 2.352723127871181713240436977837930290416,\ dddC = 310370.293289814155821882457203218819994 5,\ dddG = 322215.629414517471913296293832212785608 9\} \quad (4.1)$$

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:=100: mu:=2: y0:=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;
```

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

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

$$Rc := 0.036728412581838220293476617184035408144 72 \quad (5.2.2)$$

```
> 1/%;
```

$$27.226877768588576467079458051494458287 52 \quad (5.2.3)$$

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

$$z0 := 0.036544770518929029192009234098115231104 00 \quad (5.2.4)$$

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);
```

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

```
> 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));
```

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

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

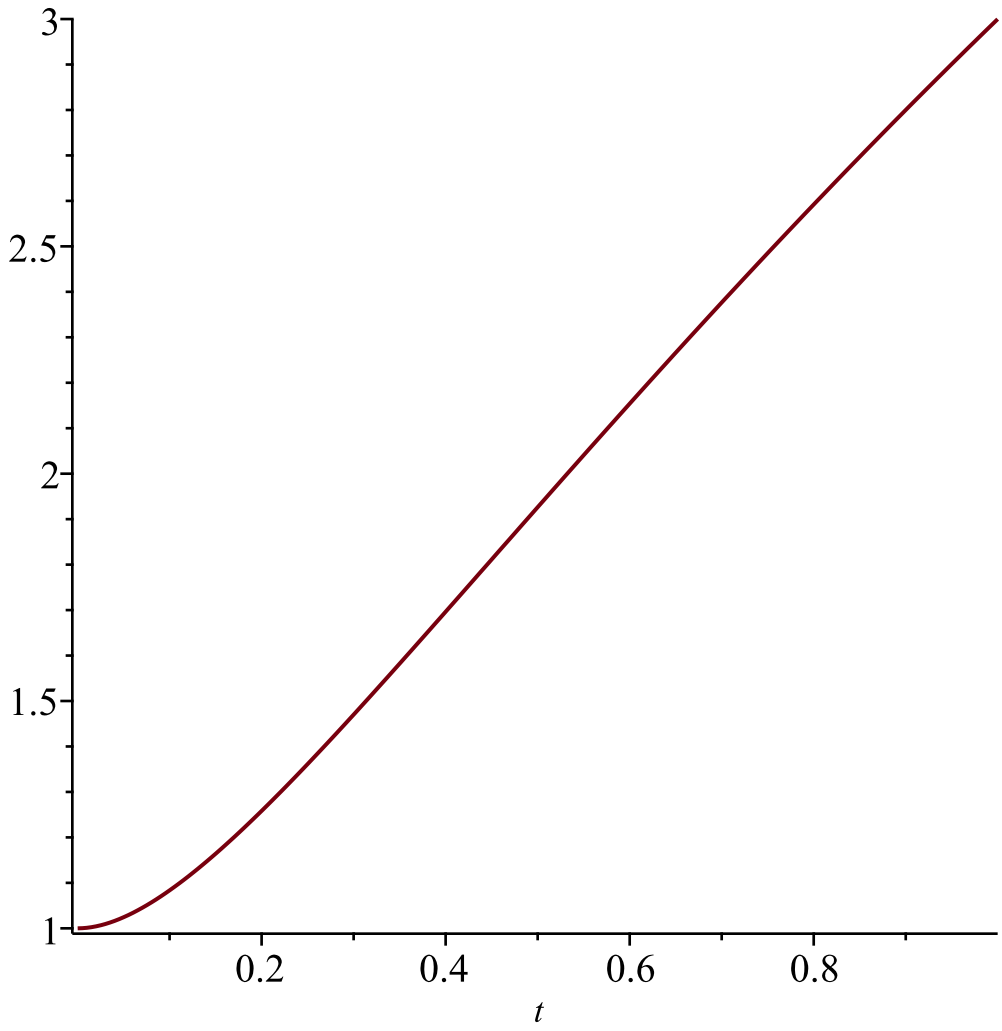
```
(3+t)*(1+2*t)*(1+3*t)^2))-1;
```

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

```
> 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
13.807868616660555822922039516667980510 08 **(5.3.4)**

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




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

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

$$x0 := 0.037992196457707622946696582954948631589 74 \tag{5.1}$$

Calculating the choose-vectors of bicolored binary trees

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

$$solutions\_networks := \{D = 1.093478486474725492111842399417696010608,\ H = 0.00206469524549284585247141356302904687085 9,\ K = 0.00206469524549284585247141356302904687085 9,\ P = 0.04779863695398693227527355784782738099750,\ S = 0.04361515427524571398409742800683958273997,\ u = 0.46825732562112923770068959141396917370 98,\ v = 2.35729858601320440235010636904540477619 6\} \tag{5.4.1}$$

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

$$u\_eval := 0.46825732562112923770068959141396917370 98$$

$$v\_eval := 2.35729858601320440235010636904540477619 6 \tag{5.4.2}$$

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

$$ch\_1\_or\_u := [0.68107952369790601564475142994744397406 46] \tag{5.4.3}$$

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

$$ch\_1\_or\_v := [0.29785852356596648452269247726807041074 28] \tag{5.4.4}$$

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

$$ch\_u\_or\_v := [0.16572219423903874574288510537990322541 90] \tag{5.4.5}$$

Calculating the choose-vectors of pointed bicolored binary trees

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);
```

$$solution\_derivate\_3\_connected := \{dxK = 0.21339174684114901055329420829854173799 07,\ dyK = 0.01826982137620513621333918972896108705372\}$$

$$solution\_bi\_derivate\_3\_connected := \{dxxK = 72.06586720994562680401670299562341420760,\ dxyK = 6.59422732406276417851230969543938450814 0,\ dyyK = 0.54409220326099477506780375516765301737 45\} \tag{5.6.1}$$

```
> 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);
```

$$K\_eval := 0.00206469524549284585247141356302904687085 9$$

$$dxK\_eval := 0.21339174684114901055329420829854173799 07$$

$$dyK\_eval := 0.01826982137620513621333918972896108705372$$

$$dxyK\_eval := 6.59422732406276417851230969543938450814 0 \tag{5.6.2}$$

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

$$ch\_K\_in\_dyK := [0.11301124422496724654164201767870832332 97] \tag{5.6.3}$$

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

$$ch\_dxK\_in\_dxyK := [0.03236038679808150597199096964197739237834] \tag{5.6.4}$$

Calculating the choose-vectors of networks

Calculating the choose-vectors of derivate networks

Calculating the choose-vectors of bi-derivate networks

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

$$solution\_planar\_graphs := \{C = 0.037248430505369045620266187796331134907 24, G = 1.037950847464415318941423252216049127410, dC = 1.039606923732873712783121705560922166006, dG = 1.079060887518410052986390325133982022943, ddC = 1.183138653548787482311219664317983257 56, ddG = 2.349838937912349978219830027637568627381, dddC = 8.203364771845075577004497014550346554, dddG = 13.510955537618490187293933662013376312 73\} \quad (5.2)$$

Calculating the choose-vectors of 2-connected planar graphs

```
> B_eval:=eval_2connected(x0,y0);dB_eval:=
  eval_derivate_2connected(x0,y0);ddB_eval:=
  eval_bi_derive_2connected(x0,y0);dddB_eval:=
  eval_tri_derivate_2connected(x0,y0);
```

$$B\_eval := 0.000731952920016455810748700553895272368 50$$
$$dB\_eval := 0.038842683760013258631145523348425252874 0$$
$$ddB\_eval := 1.050994403039639979803343196889988070416$$
$$dddB\_eval := 3.235355392329780213371171382531019136898 \quad (5.10.1)$$

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

$$ch\_xy\_in\_dB := [0.97810431154653703625570339237887034198 94] \quad (5.10.2)$$

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

$$ch\_y\_in\_ddB := [0.95147985289725977527195705394044297043 13] \quad (5.10.3)$$

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

$$ch\_nontrivialD\_or\_dD := [0.41621765075877382780851169096156827299 91] \quad (5.10.4)$$

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

$$ch\_dD\_or\_ddD := [0.19987346463953749628366439373088585802 33] \quad (5.10.5)$$

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);
```

$$C\_eval := 0.037248430505369045620266187796331134907 24$$
$$dC\_eval := 1.039606923732873712783121705560922166006$$
$$ddC\_eval := 1.183138653548787482311219664317983257 56$$
$$dddC\_eval := 8.203364771845075577004497014550346554 \quad (5.11.1)$$

```
> 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;
```

$$[0.96190202005325362308114143738582497387 39, 0.99926487602630008542563847095098742458 22, 0.99999051282576606852611843914495128442 61, 0.99999990805267483030591847829373021006 68, 0.99999999928663174795610692724084511597 90, 0.99999999999538609530149898116276019225 72, \quad (5.11.2)$$

```
    0.99999999999997441546441092385545520121 22,
    0.99999999999999875845707735441967231599 6,
    0.99999999999999999946440012561805735598 92,
    0.99999999999999999997920322877831513107 0,
    0.99999999999999999999992658508005643097 1,
    0.99999999999999999999999762423263214172,
    0.99999999999999999999999999290296595758,
    0.99999999999999999999999999998031308897,
    0.99999999999999999999999999999994902872,
    0.99999999999999999999999999999999987623,
    0.99999999999999999999999999999999999967,
    0.99999999999999999999999999999999999995,
    0.99999999999999999999999999999999999995,
    0.99999999999999999999999999999999999995,
    0.99999999999999999999999999999999999995,
    0.99999999999999999999999999999999999995]
> ch_dC_or_ddC:=[evalf(dC_eval/(z0*ddC_eval+dC_eval))];
```

$$ch\_dC\_or\_ddC := [0.96007041416376685217231934685416949410 94] \quad (5.11.3)$$

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

$$ch\_2ddC\_or\_dddC := [0.88755344888289363094399652191006473564 44] \quad (5.11.4)$$

```
> 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]];
```

$$choose\_vector\_dddC\_Bernoulli := [0.35509835952770902048963112857201991830 43, 0.48076327674300568858553573749402670243 42, 0.16413836372928529092483313393395337953 60]$$

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

$$choose\_vector\_dddC := [0.35509835952770902048963112857201991830 43, 0.83586163627071470907516686606604662073 85] \quad (5.11.5)$$

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);
```

$$G\_eval := 1.037950847464415318941423252216049127410$$

$$dG\_eval := 1.079060887518410052986390325133982022943$$

$$ddG\_eval := 2.349838937912349978219830027637568627381$$

$$dddG\_eval := 13.510955537618490187293933662013376312 73 \quad (5.12.1)$$

```
> 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;
```

$$[0.96343675853521916390417976244594921678 68,\ 0.99932326568183629264194968216228236636 74,\ 0.99999162371560189150390996792361551351 27,\ 0.99999992214485969917300148982607211699 63,\ 0.99999999942072607746554893589426712148 75,\ 0.99999999999640702517235179872296120496 95,\ 0.99999999999980893801338771736999795498 2,\ 0.99999999999999991108666193456748960424 05,\ 0.99999999999999999632165779643165123815 7,\ 0.99999999999999999998630340369408336754 0,\ 0.99999999999999999999953633416975432328,\ 0.99999999999999999999999856110904111241,\ 0.99999999999999999999999995878043256 86,\ 0.99999999999999999999999999989035062 06,\ 0.99999999999999999999999999999972775 81,\ 0.99999999999999999999999999999999936 65,\ 0.99999999999999999999999999999999999 88,\ 1.00000000000000000000000000000000000 00,\ 1.00000000000000000000000000000000000 00,\ 1.00000000000000000000000000000000000 00,\ 1.00000000000000000000000000000000000 00]$$ (5.12.2)

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

$$choose\_vector\_ddG := [0.52260593196650725495256243899738497223 10]$$ (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]];
```

$$choose\_vector\_dddG\_Bernoulli := [0.63020630874618130252158298587618721293 67,\ 0.18898421240467277396693184348173258614 78,\ 0.18080947884914592351148517064208020091 53]$$

$$choose\_vector\_dddG := [0.63020630874618130252158298587618721293 67,\ 0.81919052115085407648851482935791979908 45]$$ (5.12.4)

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;

                [0.68107952369790601564475142994744397406 46]
                [0.29785852356596648452269247726807041074 28]
                [0.16572219423903874574288510537990322541 90]
                [0.23747240472499782031031511012907581387 49]
[0.10429082361681012978795865717642333003 77,
    0.55214541180840506489397932858821166501 87]
                                [0.5]
                [0.22576500850646120628895898552749928784 23]
[0.06367579431384127520915197697375162482927,
    0.53183789715692063760457598848687581241 48]
[0.04337710064133745054430404177895476405318,
    0.52168855032066872527215202088947738202 65]
                [0.11301124422496724654164201767870832332 97]
                [0.03236038679808150597199096964197739237834]
                [0.13001418656669182512786022459244070058 21]
                [0.15058380535487698618694197235053446080 83]
[0.46657959408701256213966406465848037115 38,
    0.97791261579688612503646191603060220097 17]
[0.91451273378400742711259959255190774735 82,
    0.95439934775467362414118734059064476495 24,
    0.99811180990661349520031818951101094846 57]
                [0.97783631952425745633475078020359911310 94]
[0.95250493024171505005929617550685637894 71,
    0.99803336759922143613116943185562863106 41]
```

[0.954800743278379707394139844663889140463 8]
[0.955347768281996121055136083212182318805 1,
0.998987910577091114281767161697636145279 6,
0.999984648143642893072296778332948722419 2,
0.999999825084326810535888103331480560227 7,
0.999999998404418467167063049064929275367 6,
0.999999999987865608326168417110912528416 9,
0.999999999999920879514867017452904961701 5,
0.999999999999999548511128652756815783586 1,
0.999999999999999997709617550137439882066 0,
0.999999999999999999989541925445828503569 3,
0.999999999999999999999565856841036092091,
0.999999999999999999999998347852394297564,
0.999999999999999999999999994196087457135,
0.999999999999999999999999999981066854285,
0.999999999999999999999999999999942353452,
0.999999999999999999999999999999999835449,
0.999999999999999999999999999999999999560,
1.000000000000000000000000000000000000000,
1.000000000000000000000000000000000000000,
1.000000000000000000000000000000000000000,
1.000000000000000000000000000000000000000]
[0.977333956580252874206404297255756340371 3,
0.999656190614299866558674861548098210023 1,
0.999996082711513858382934977343744393469 9,
0.999999964266477364230040597789118315662 6,
0.999999997282466921146302026589873473598,
0.999999999982280732207819079317633936615,
0.999999999999898887725433619957431350733,
0.999999999999999487062043320205912551262,
0.999999999999999997657883122120685054956,
0.999999999999999999990277234927344954539,
0.999999999999999999999962999663350840146,
0.999999999999999999999999870019653675554,
0.999999999999999999999999995759865747 49,
0.999999999999999999999999999998708988413,
0.999999999999999999999999999999996314829,
0.999999999999999999999999999999999990134,
1.000000000000000000000000000000000000001,
1.000000000000000000000000000000000000004,
1.000000000000000000000000000000000000004,
1.000000000000000000000000000000000000004,
1.000000000000000000000000000000000000004]
[0.984831521790847731544024426481282468449 5,
0.999827173697076358976673121512179478548 7,
0.999998423477711833986188282456345091522 1,
0.999999988010553811583511956797641277978 2,
0.999999999218246102151929695770085126295,
0.999999999995539041698019119745467014257,
0.999999999999977369762018857162724039142,
0.999999999999999896666847298815221175035 3,
0.999999999999999999571042687397996802004,

0.99999999999999999999983675873215294223 42,
0.99999999999999999999999426541527705856 0,
0.99999999999999999999999999812930109873 20,
0.99999999999999999999999999943042041949,
0.99999999999999999999999999999837416091,
0.99999999999999999999999999999999566350,
1.00000000000000000000000000000000000206,
1.00000000000000000000000000000000000317,
1.00000000000000000000000000000000000317,
1.00000000000000000000000000000000000317,
1.00000000000000000000000000000000000317,
1.00000000000000000000000000000000000317]
[0.39756970098378115947748268192048522785 79,
0.91989581684278309894991464031439406845 56]
[0.06295035340164952140325678541740341992929,
0.89967390907313288972798715019279275171 68]
[0.95725636689200371355629979627595387367 93]
[0.77192425343945179324274319108366968252 97]
[0.86703161629149680593973535103200608547 39]
[0.83230361588373536424652854028519743930 34]
[0.11075936959758734178998711415038739402 15,
0.64055859934611062047015028797129761298 40]
[0.33353657742373343603560486815154576157 31,
0.44642708851822419025384963779241921569 89,
0.45996305700396522092772133139779813937 70,
0.63988045331439804069194626094210967889 60]
[0.92898889010517227281955601010955813197 93,
0.94375373146037243980331228636452599932 85,
0.98523515864479983301624372374503213265 19]
[0.55137192991609524754807095499183368457 48,
0.89959431172810898944610853176093043202 00,
0.95042268414978607869630796123707692648 22]
[0.59578837452441921727886962924750488285 74]
[0.23555760986599423556654242838296475867 94]
[0.97810431154653703625570339237887034198 94]
[0.95147985289725977527195705394044297043 13]
[0.41621765075877382780851169096156827299 91]
[0.19987346463953749628366439373088585802 33]
[0.96190202005325362308114143738582497387 39,
0.99926487602630008542563847095098742458 22,
0.99999051282576606852611843914495128442 61,
0.99999990805267483030591847829373021006 68,
0.99999999928663174795610692724084511597 90,
0.99999999995386095301498981162760192257 2,
0.99999999999997441546441092385545520121 22,
0.99999999999999987584570773544196723159 96,
0.99999999999999999946440012561805735598 92,
0.99999999999999999997920322877831513107 0,
0.99999999999999999999999265850800564309 71,
0.99999999999999999999999976242326321417 2,
0.99999999999999999999999999290296595758,
0.99999999999999999999999999998031308897,

```
    0.999999999999999999999999999999994902872,
    0.999999999999999999999999999999987623,
    0.999999999999999999999999999999999999967,
    0.999999999999999999999999999999999999995,
    0.999999999999999999999999999999999999995,
    0.999999999999999999999999999999999999995,
    0.999999999999999999999999999999999999995,
    0.999999999999999999999999999999999999995]
                [0.960070414163766852172319346854169494109 4]
                [0.887553448882893630943996521910064735644 4]
[0.355098359527709020489631128572019918304 3,
    0.835861636270714709075166866066046620738 5]
[0.963436758535219163904179762445949216786 8,
    0.999323265681836292641949682162282366367 4,
    0.999991623715601891503909967923615513512 7,
    0.999999922144859699173001489826072116996 3,
    0.999999999420726077465548935894267121487 5,
    0.999999999996407025172351798722961204969 5,
    0.999999999999980893801338771736999795498 2,
    0.999999999999999911086661934567489604240 5,
    0.999999999999999996321657796431651238157,
    0.999999999999999999998630340369408336754 0,
    0.999999999999999999999953633416975432328,
    0.999999999999999999999999856110904111241,
    0.999999999999999999999999999587804325686,
    0.999999999999999999999999999998903506206,
    0.999999999999999999999999999999997277581,
    0.999999999999999999999999999999999993665,
    0.999999999999999999999999999999999999988,
    1.000000000000000000000000000000000000000,
    1.000000000000000000000000000000000000000,
    1.000000000000000000000000000000000000000,
    1.000000000000000000000000000000000000000]
                [0.522605931966507254952562438997384972231 0]
[0.630206308746181302521582985876187212936 7,                    (5.13.1)
    0.819190521150854076488514829357919799084 5]
```

```
> eval_C(1,1);
                0.95653495653442361453871503387427915874 18                    (1)
>
```