1 Question 1

Statement 1 ▶ Two important facts

$$N'(x) = \frac{1}{\sqrt{2\pi}} \exp\left(\frac{x^2}{2}\right)$$
$$SN'(d_1) = \exp\left(-r(T-t)\right) EN'(d_2)$$

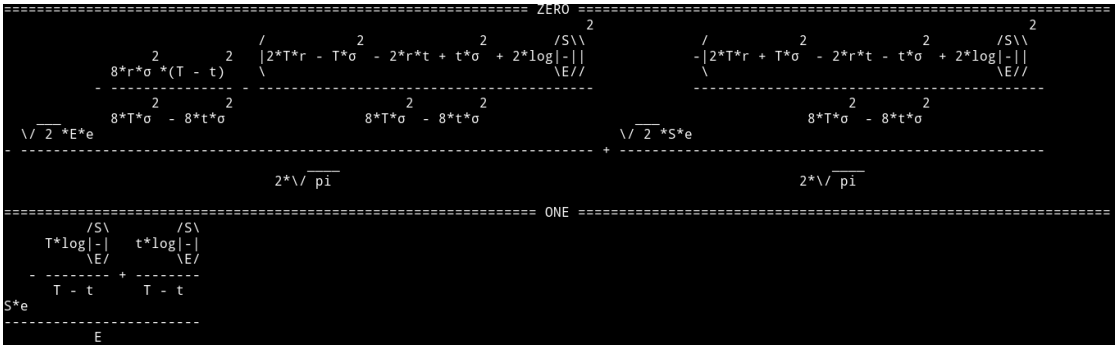

Figure 1: 公式符号推导结果

首先, 我们尝试使用符号推导解决问题, 基于 sympy 库的程序请参见附录 A. 运行结果如图 1. 可以看出, 如果不经变换使用符号计算, 是得不到答案的; 经过如式 (1)变换:

$$\log\left(\frac{SN'(d_1)}{\exp\left(-r(T-t)\right) EN'(d_2)}\right), \tag{1}$$

符号简化到如式 (2), 此时符号推导已经进行不下去了, 通过人为演算可以发现式 (2)恒等于 1, 因此等式成立.

$$S\exp\left(-\frac{T\log\left(\frac{S}{E}\right)}{T-t} + \frac{t\log\left(\frac{S}{E}\right)}{T-t}\right), \tag{2}$$

最后, 我们采用人工推导, 如式 (3).

$$\begin{aligned}
\log\left(\frac{SN'(d_1)}{\exp\left(-r(T-t)\right)EN'(d_2)}\right) &= \log\left(\frac{S\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{d_1^2}{2}\right)}{\exp\left(-r(T-t)\right)E\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{d_2^2}{2}\right)}\right) \\
&= \log\left(\frac{S}{E}\right) + r(T-t) - \frac{1}{2}\left(d_1^2 - d_2^2\right) \\
&= \log\left(\frac{S}{E}\right) + r(T-t) - \frac{1}{2}(d_1+d_2)(d_1-d_2) \\
&= \log\left(\frac{S}{E}\right) + r(T-t) - \frac{1}{2}\frac{2\log\left(\frac{S}{E}\right)+2r(T-t)}{\sigma\sqrt{T-t}}\frac{\sigma^2(T-t)}{\sigma\sqrt{T-t}} \\
&= \log\left(\frac{S}{E}\right) + r(T-t) - \frac{1}{2}\left(2\log\left(\frac{S}{E}\right)+2r(T-t)\right) \\
&= 0
\end{aligned} \tag{3}$$

2 Question 2

Statement 2 ▶ Symbolic Derivation

1. Adapt function `lect7_1.m` to return more Greeks.
2. Investigate the use of MATLAB's symbolic toolbox to confirm the results in this lecture.

```
greeks('S', 'E', 'r', 'sigma', 'tau', to_simple=True, to_latex=True)
```

代码详见附录 B, 运行结果如式 4与式 5 (可能不美观).

$$
\begin{aligned}
C &= S\left(\frac{\operatorname{erf}\left(\frac{\sqrt{2}\left(\ln\left(\frac{S}{E}\right)+\tau\left(\frac{\sigma^2}{2}+r\right)\right)}{2\sigma\sqrt{\tau}}\right)}{2}+\frac{1}{2}\right)+E\,\mathrm{e}^{-r\tau}\left(\frac{\operatorname{erf}\left(\frac{\sqrt{2}\left(\sigma\sqrt{\tau}-\frac{\ln\left(\frac{S}{E}\right)+\tau\left(\frac{\sigma^2}{2}+r\right)}{\sigma\sqrt{\tau}}\right)}{2}\right)}{2}-\frac{1}{2}\right)\\
C\delta &= \frac{\operatorname{erf}\left(\frac{\sqrt{2}\left(\ln\left(\frac{S}{E}\right)+\tau\left(\frac{\sigma^2}{2}+r\right)\right)}{2\sigma\sqrt{\tau}}\right)}{2}+\frac{1}{2}\\
C\nu &= \frac{7186705221432913\,S\sqrt{\tau}\,\mathrm{e}^{-\frac{\left(\ln\left(\frac{S}{E}\right)+\tau\left(\frac{\sigma^2}{2}+r\right)\right)^2}{2\sigma^2\tau}}}{18014398509481984}\\
C\theta &= E\,r\,\mathrm{e}^{-r\tau}\left(\frac{\operatorname{erf}\left(\frac{\sqrt{2}\left(\sigma\sqrt{\tau}-\frac{\ln\left(\frac{S}{E}\right)+\tau\left(\frac{\sigma^2}{2}+r\right)}{\sigma\sqrt{\tau}}\right)}{2}\right)}{2}-\frac{1}{2}\right)-\frac{7186705221432913\,S\,\sigma\,\mathrm{e}^{-\frac{\left(\ln\left(\frac{S}{E}\right)+\tau\left(\frac{\sigma^2}{2}+r\right)\right)^2}{2\sigma^2\tau}}}{36028797018963968\sqrt{\tau}}\\
C\rho &= E\,\tau\,\mathrm{e}^{-r\tau}\left(\frac{\operatorname{erf}\left(\frac{\sqrt{2}\left(-\tau\sigma^2+2\ln\left(\frac{S}{E}\right)+2r\tau\right)}{4\sigma\sqrt{\tau}}\right)}{2}+\frac{1}{2}\right)\\
C\gamma &= \frac{7186705221432913\,\mathrm{e}^{-\frac{\left(\ln\left(\frac{S}{E}\right)+\tau\left(\frac{\sigma^2}{2}+r\right)\right)^2}{2\sigma^2\tau}}}{18014398509481984\,S\,\sigma\sqrt{\tau}}
\end{aligned}
\tag{4}
$$

$$P = \frac{E\,\mathrm{e}^{-r\,\tau}}{2} - \frac{S}{2} + \frac{S\,\mathrm{erf}\left(\frac{\sqrt{2}\left(\ln\left(\frac{S}{E}\right)+\tau\left(\frac{\sigma^2}{2}+r\right)\right)}{2\,\sigma\sqrt{\tau}}\right)}{2} + \frac{E\,\mathrm{erf}\left(\frac{\sqrt{2}\left(\sigma\sqrt{\tau}-\frac{\ln\left(\frac{S}{E}\right)+\tau\left(\frac{\sigma^2}{2}+r\right)}{\sigma\sqrt{\tau}}\right)}{2}\right)\mathrm{e}^{-r\,\tau}}{2}$$

$$P\delta = \frac{\mathrm{erf}\left(\frac{\sqrt{2}\left(\ln\left(\frac{S}{E}\right)+\tau\left(\frac{\sigma^2}{2}+r\right)\right)}{2\,\sigma\sqrt{\tau}}\right)}{2} + \frac{1}{2}$$

$$Pv = \frac{7186705221432913\,S\sqrt{\tau}\,\mathrm{e}^{-\frac{\left(\ln\left(\frac{S}{E}\right)+\tau\left(\frac{\sigma^2}{2}+r\right)\right)^2}{2\,\sigma^2\,\tau}}}{18014398509481984}$$

$$P\theta = E\,r\,\mathrm{e}^{-r\,\tau}\left(\frac{\mathrm{erf}\left(\frac{\sqrt{2}\left(\sigma\sqrt{\tau}-\frac{\ln\left(\frac{S}{E}\right)+\tau\left(\frac{\sigma^2}{2}+r\right)}{\sigma\sqrt{\tau}}\right)}{2}\right)}{2} - \frac{1}{2}\right) - \frac{7186705221432913\,S\,\sigma\,\mathrm{e}^{-\frac{\left(\ln\left(\frac{S}{E}\right)+\tau\left(\frac{\sigma^2}{2}+r\right)\right)^2}{2\,\sigma^2\,\tau}}}{36028797018963968\sqrt{\tau}}$$

$$P\rho = E\,\tau\,\mathrm{e}^{-r\,\tau}\left(\frac{\mathrm{erf}\left(\frac{\sqrt{2}\left(-\tau\,\sigma^2+2\,\ln\left(\frac{S}{E}\right)+2\,r\,\tau\right)}{4\,\sigma\sqrt{\tau}}\right)}{2} + \frac{1}{2}\right)$$

$$P\gamma = \frac{7186705221432913\,\mathrm{e}^{-\frac{\left(\ln\left(\frac{S}{E}\right)+\tau\left(\frac{\sigma^2}{2}+r\right)\right)^2}{2\,\sigma^2\,\tau}}}{18014398509481984\,S\,\sigma\sqrt{\tau}}$$

(5)

A 附录: 问题一 Python 代码

8-a.py

```python
#!/usr/bin/python3
# -*- encoding: utf-8 -*-
'''
@File        : 8-a.py
@Time        : 2019/11/14
@Author      : Iydon Liang
@Contact     : liangiydon@gmail.com
@Docstring   : <no docstring>
'''

from os import get_terminal_size
columns = get_terminal_size().columns

from sympy import symbols, integrate, diff, exp, sqrt, log
from sympy import oo, pi as π
from sympy import pretty_print
hw7a = __import__('7-a')


# Symbols
S, E, r, σ, T, t = (getattr(hw7a, _) for _ in ('S', 'E', 'r', 'σ', 'T', 't'))
d1, d2 = hw7a.d1, hw7a.d2
_ = symbols('_')

N:_ = integrate(exp(-_**2/2)/(sqrt(2*π)), (_, -oo, _))
dN:_ = diff(N, _)



if __name__ == "__main__":
    # Results in Different Format
    ZERO = S*dN.subs(_, d1) - exp(-r*(T-t))*E*dN.
```

```
        subs(_, d2)
33      ONE = (S*dN.subs(_, d1)) / (exp(-r*(T-t))*E*dN.
        subs(_, d2))
34
35      # Simplify Result
36      print(f'{"ZERO ":=^{columns}}')
37      pretty_print(ZERO.simplify(), use_unicode=False)
38
39      print()
40
41      print(f'{"ONE ":=^{columns}}')
42      pretty_print(ONE.simplify(), use_unicode=False)
```

B 附录: 问题二 Python 代码

8-b.py

```
1  #!/usr/bin/python3
2  # -*- encoding: utf-8 -*-
   '''
3
4  @File       : 8-b.py
5  @Time       : 2019/11/14
6  @Author     : Iydon Liang
7  @Contact    : liangiydon@gmail.com
8  @Docstring  : <no docstring>
   '''
9
10
11 from sympy import diff, exp
12 hw8a = __import__('8-a')
13
14
15 # Util
16 class Simplify:
17     _SIMPLIFY = 'simplify'
18     def __init__(self, lazy=False):
19         self._lazy = lazy
20     def __ror__(self, value):
```

```
21          if self._lazy:
22              return value
23          _simplify = getattr(value, self._SIMPLIFY,
                None)
24          if _simplify is not None:
25              try:
26                  return _simplify()
27              finally:
28                  print('Formula simplified.')
29  simplify = Simplify(lazy=False)
30
31  # Symbols
32  S, E, r, σ, T, t, d1, d2, _, N = (getattr(hw8a, _)
33      for _ in ('S', 'E', 'r', 'σ', 'T', 't', 'd1', '
            d2', '_', 'N'))
34
35
36  if __name__ == "__main__":
37      # Derivation
38      C = S*N.subs(_, d1) - E*exp(-r*(T-t)*N.subs(_,
            d2)) | simplify
39      Δ = diff(C, S) | simplify
40      Cₜ = diff(C, t) | simplify
41      Γ = diff(C, S, 2) | simplify
42
43      PDE = Cₜ + r*S*Δ + σ**2*S**2*Γ/2 - r*C |
            simplify
```

8-b.py

```
1  #!/usr/bin/python3
2  # -*- encoding: utf-8 -*-
   '''
3
4  @File       : 8-b_MATLAB.py
5  @Time       : 2019/11/15
6  @Author     : Iydon Liang
7  @Contact    : liangiydon@gmail.com
8  @Docstring  : <no docstring>
```

```
    '''

from math import pi as π

import matlab
import matlab.engine
if 'engine' not in locals():
    engine = matlab.engine.start_matlab()


def greeks(S, E, r, σ, τ, to_simple=False, to_str=
    False, to_latex=False):
    '''Greeks.

    :Argument:
        - S: [str, float], asset price @ time t
        - E: [str, float], exercise price
        - r: [str, float], interest rate
        - σ: [str, float], volatility
        - τ: [str, float], time to expiry (T-t)
        - to_simple: bool, wheather to simplify
          result
        - to_str: bool, wheather to convert result
          to str
        - to_latex: bool, wheather to convert result
          to latex

    :Output:
        - C, call value
        - Cδ, δ value of call
        - Cν, ν value of call
        - Cθ, θ value of call
        - Cρ, ρ value of call
        - Cγ, γ value of call
        - P, put value
        - Cδ, δ value of put
        - Cν, ν value of put
        - Cθ, θ value of put
        - Cρ, ρ value of put
```

```
44          - Cγ, γ value of put
45
46      :Example:
47          >>> S=1.0; E=1.5; r=0.05; σ=0.2; τ=1.0;
48          >>> greeks(S, E, r, σ, τ)
49      '''
50      _f = lambda x: engine.str2sym(str(x))
51      S, E, r, σ, τ = _f(S), _f(E), _f(r), _f(σ), _f
            (τ)
52      乘方 = engine.power
53      逆 = engine.inv
54      乘 = engine.times
55      除 = lambda x, y: 乘(x, 逆(y))
56      加 = engine.plus
57      减 = lambda x, y: 加(x, 乘(y, -1.))
58      log, sqrt, erf, exp = engine.log, engine.sqrt,
            engine.erf, engine.exp
59      d1 = 除(加(log(除(S, E)), 乘(加(r, 乘(1/2, 乘方
            (σ, 2))), τ)), 乘(σ, sqrt(τ)))
60      d2 = 减(d1, 乘(σ, sqrt(τ)))
61      Nd1 = 乘(1/2, 加(1, erf(除(d1, sqrt(2.)))))
62      Nd2 = 乘(1/2, 加(1, erf(除(d2, sqrt(2.)))))
63      Np1 = 除(exp(乘(-1/2, 乘方(d1, 2))), sqrt(乘(2.,
            π)))
64
65      C = 减(乘(S, Nd1), 乘(乘(E, Nd2), exp(乘(减(0, r
            ), τ))))
66      Cδ = Nd1
67      Cν = 乘(乘(S, Np1), sqrt(τ))
68      Cθ = 减(除(乘(乘(减(0, S), σ), Np1), 乘(2., sqrt
            (τ))),
69          乘(乘(乘(r, E), Nd2), exp(乘(减(0, r), τ))))
70      Cρ = 乘(乘(乘(E, τ), Nd2), exp(乘(减(0, r), τ)))
71      Cγ = 除(Np1, 乘(乘(S, σ), sqrt(τ)))
72
73      P = 加(减(C, S), 乘(E, exp(乘(减(0, r), τ))))
74      Pδ = 减(Cδ, -1.)
75      Pν = Cν
76      Pθ = 加(除(乘(乘(减(0, S), σ), Np1), 乘(2., sqrt
```

```
            (τ))),
77              乘(乘(乘(r, E), Nd2), exp(乘(减(0, r), τ))))
78          Pρ = 减(Cρ, -1.)
79          Pγ = Cγ
80
81          result = C, Cδ, Cν, Cθ, Cρ, Cγ, P, Cδ, Cν, Cθ, C
               ρ, Cγ
82          if to_simple:
83              result = tuple(engine.simplify(r) for r in
                   result)
84          if to_str and not to_latex:
85              result = tuple(engine.char(r) for r in
                   result)
86          if to_latex:
87              result = tuple(engine.latex(r) for r in
                   result)
88          return result
89
90
91  if __name__ == "__main__":
92      S='S'; E='E'; r='r'; σ='sigma'; τ='tau';
93      result = greeks(S, E, r, σ, τ, to_simple=True,
            to_latex=True)
```