



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

SYSTÉM ŘÍZENÍ ZÁSOB

INVENTORY MANAGEMENT SYSTEM

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE

AUTHOR

Bc. LUDĚK REIF

VEDOUCÍ PRÁCE

SUPERVISOR

prof. RNDr. Ing. MILOŠ ŠEDA, Ph.D.

BRNO 2015

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky

Akademický rok: 2014/2015

ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Bc. Luděk Reif

který/která studuje v **magisterském navazujícím studijním programu**

obor: **Aplikovaná informatika a řízení (3902T001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Systém řízení zásob

v anglickém jazyce:

Inventory Management System

Stručná charakteristika problematiky úkolu:

Řízení zásob na skladě patří mezi důležité funkce v logistice. Základním problémem je to, že poptávka po skladovaných položkách má stochastický charakter. Při řízení skladových zásob je třeba zohlednit náklady na udržování zásob a ztráty z předčasného vyčerpání zásoby v Kč/jednotku nedodaného zboží.

Cíle diplomové práce:

1. Proveďte klasifikaci systémů řízení skladových zásob (s pevnou velikostí doplňovací objednávky a proměnným objednacím termínem, s proměnnou velikostí doplňovací objednávky a pevným objednacím termínem, ...).
2. Implementujte systém řízení skladových zásob při náhodných výkyvech poptávky včetně generování dat a grafického znázornění simulačních výpočtů.

Seznam odborné literatury:

- [1] Bartmann, D. and Beckmann, M. J.: Inventory Control. Models and Methods. Springer-Verlag, Berlin, 1992.
- [2] Hrubina, K., Jadlovská, A., Hrehová, S.: Algoritmy optimalizačních metod s využitím programových systémů. Technická univerzita v Košiciach, Prešov-Košice, 2005.
- [3] Jablonský, J.: Operační výzkum. Vysoká škola ekonomická, Fakulta informatiky a statistiky, Praha, 2001.
- [4] Klvaňa, J.: Modelování 20. České vysoké učení technické, Fakulta stavební, Praha, 2005.
- [5] Peltan, K., Májek, V.: Operační výzkum ve vojenství. Univerzita obrany, Brno, 2008.

Vedoucí diplomové práce: prof. RNDr. Ing. Miloš Šeda, Ph.D.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2014/2015.

V Brně, dne 17.10.2014

L.S.

Ing. Jan Roupec, Ph.D.
Ředitel ústavu

doc. Ing. Jaroslav Katolický, Ph.D.
Děkan fakulty

Abstrakt

Tato diplomová práce se zabývá charakteristikou a implementací systémů řízení zásob. Teoretická část obsahuje jednotlivé systémy, a to jak systémy deterministické, tak stochastické. Pro implementaci je využito několika příkladů, které jsou pak algoritmizovány. U deterministických systémů jsou využity triviální vzorce. U systémů stochastických je začleněn prvek náhody, který je simulován generátorem pseudonáhodných čísel. Výsledkem je pak komplexní počítačový program, ve kterém je možné vyhodnocovat všechny zde uvedené modely s vlastními vstupními hodnotami. Program poskytuje jak početní funkci, tak grafický výstup.

Abstract

My thesis involves in characteristics and implementation of inventory management system. Theoretical part contains particular systems, both deterministic and stochastic. Several examples are used for implementation, which are then algorithmized. Trivial equations are used in deterministic systems. In stochastic systems I used an element of randomness, which is simulated by pseudo random number generator. The result is a complex computer program, in which there is possible to evaluate all models mentioned here, with their own input values. The program then yields both numerical values and graphical output.

Klíčová slova

řízení zásob, logistika, stochastický, deterministický, implementace

Keywords

inventory management, logistics, stochastic, deterministic, implementation

Prohlášení o originalitě

Prohlašuji, že jsem diplomovou práci Systém řízení zásob vypracoval samostatně pod vedením prof. RNDr. Ing. Miloše Šedy, Ph.D. s použitím materiálů uvedených v seznamu literatury.

Bc. Luděk Reif, Brno 2015

Bibliografická citace

REIF, L. *Systém řízení zásob*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2015. 78 s. Vedoucí diplomové práce prof. RNDr. Ing. Miloš Šeda, Ph.D..

Poděkování

Děkuji svému vedoucímu práce prof. RNDr. Ing. Miloši Šedovi za vedení a rady, které mi pomohly k vytvoření diplomové práce.

Bc. Luděk Reif

Obsah

1	Úvod	13
2	Teorie zásob	15
2.1	Kriteriální funkce	15
2.2	Funkce zásob	16
2.3	Druhy nákladů	16
3	Deterministické modely řízení zásob	19
3.1	Optimální velikost objednávky	19
3.1.1	Doplnění o přírážku k ceně	20
3.1.2	Přírážka při různém množství	20
3.1.3	Doba potřebná k vyřízení objednávky	20
3.2	Optimální velikost objednávky při nespojitém množství	20
3.3	Optimální velikost objednávky při množstevních rabatech	21
3.3.1	Sleva A	21
3.3.2	Sleva B	23
4	Stochastické modely řízení zásob	25
4.1	Charakteristiky systémů zásob	25
4.1.1	Řiditelné proměnné	25
4.1.2	Neřiditelné proměnné	25
4.2	Klasifikace stochastických systémů řízení zásob	26
4.2.1	Jednorázová objednávka	26
4.2.2	Modely řízení zásob s náhodnou a proměnlivou spotřebou v čase	27
4.2.2.1	Q-systém	28
4.2.2.2	P-systém	31
5	Příklady	33
5.1	Příklad s optimální velikostí objednávky	33
5.2	Příklad s optimální velikostí objednávky při nespojitém množství	34
5.3	Příklad s dobou potřebnou pro vyřízení objednávky	35
5.4	Příklad s optimální velikostí objednávky při nespojitém objednacím množství	35
5.5	Sleva typu A	37
5.6	Sleva typu B	39
5.7	Q-systém	40
5.8	P-systém	42
6	Použité řešení	45
6.1	Python	45
6.2	Knihovny třetích stran	45
7	Program	47
7.1	Popis programu	47
7.2	Zdrojový kód	51
8	Závěr	69
9	Použitá literatura	71
	Seznamy	73
	Seznam grafů	73
	Seznam tabulek	73
	Seznam obrázků	73
	Přílohy	75

1 ÚVOD

Proces optimalizace skladových zásob je oblast, která se s příchodem výkonných počítačů dostává do stavu, kdy proces nejenom plánování zastupuje z velké části právě výpočetní technika. Samozřejmě jsou nutné expertní znalosti a znalosti o prodeji, výrobě či jiném procesu ovlivňujícím skladové zásoby. Ovšem samotné zpracování dat a jejich analýza je softwarovou záležitostí. Samotné rozhodnutí je pak na uvážení managementu a jako podklad pro co nejlepší rozhodnutí mu slouží právě výsledné analýzy.

Diplomová práce se zabývá tématem analýzy skladových zásob. Modely řízení zásob jsou v práci rozděleny na deterministické a stochastické, tedy tak, jak jsou běžně rozděleny v literatuře. K oběma modelům je uvedena nezbytná teorie, která je popisuje. Z deterministických systémů je zde uvedeno několik modelových příkladů. Ze systémů stochastických se zde nacházejí dva hlavní systémy, a to sice Q-systém a P-systém. Je tedy provedena jasná klasifikace.

Dále jsou od každého z modelů, jak deterministických, tak stochastických, vypočteny ukázkové příklady, na kterých je posléze založena implementace. Každý příklad je rozebrán na jednotlivé vzorce, případně jsou objasněny iterační metody výpočtu.

Samotná implementace je provedena v programovacím jazyku Python, který je doplněn o moduly třetích stran. Program pro každý příklad realizuje jednak výpočetní část. To znamená, že se postará o zpracování zadaných dat, která jsou různá pro různé typy příkladů, dopočítá žádané hodnoty a na jejich základě vygeneruje simulovaný průběh. Jsou generována data, která se postupně ukládají a ve výsledku jsou zobrazeny v podobě grafu. Ke grafu jsou navíc přidány výpočty ve formě matematického zápisu, relevantního pro daný příklad.

V případě deterministických modelů je průběh vždy ideální a to se také odráží na výsledcích, kdy má generování dat vždy stejnou podobu. U modelů stochastických tomu tak přirozeně není, proto je použito normální pravděpodobnostního rozdělení, podle kterého jsou pseudonáhodně generována výsledná data.

2 TEORIE ZÁSOB

Teorie zásob vytváří modely za účelem hledání způsobu nakládání (doplňování, udržování, čerpání) se zásobami tak, aby byla zajištěna jejich ekonomicky efektivní funkce v procesu tržní reprodukce. Patří k tradičním oblastem logistiky. Jejím úkolem je minimalizace nákladů a optimální řízení zásob.

Zásoby v procesu vystupují na dvou místech:

- na skladě zdrojů (zde je uložen materiál a suroviny potřebné pro výrobu)
- na skladě výrobků (zde jsou hotové výrobky nebo polotovary pro další výrobu)

Při volně optimální strategii je nutné určit minimum kritériální funkce, vyjádřenou náklady na zacházení se zásobami. Pro potřeby práce bude minimum kritériální funkce vždy rozhodující.

2.1 Kritériální funkce

Ve většině případů se určuje kritériální funkce tak, aby bylo dosaženo minimum očekávaných celkových nákladů. Ty vznikají ve vztahu k zásobám při jejich pořizování, skladování a čerpání. Tyto náklady jsou závislé na úrovni zásob a na způsobu jejich vytváření či doplňování. Je možné určit kritériální funkci tak, aby sledovala maximum očekávaného zisku. To je výhodné, pokud výše zásob bude ovlivňovat např. poptávku po určitém výrobku. Za pomoci kritériální funkce se určí strategie řízení systému zásob. Díky této strategii je možné nalézt více či méně přesnou odpověď na otázku, kdy a kolik výrobků objednat či vyrobit ve vztahu ke skladovým zásobám. Odpovědí je v podstatě nalezení optima mezi dvěma krajními situacemi. První situace nastává při nedostatečné výši zásob, kdy jsou náklady na skladování minimální, avšak při výpadku výrobního procesu nebo při zvýšení poptávky dochází ke ztrátám. V druhém případě je výše zásob extrémně vysoká, což sice pokryje jak výpadek výrobního procesu, tak zvýšenou poptávku, ovšem náklady na udržení zásob jsou zbytečně velké a v zásobách jsou zbytečně vázány značné prostředky. Navíc zásoby mohou zastarávat nebo se dokonce samovolně znehodnocovat. [1]

V případě deterministických modelů se jedná o náklady na udržování zásob (N_1) a o náklady za pořizování zásob (N_2):

$$N_1 = \bar{x}Tn_sc \quad N_2 = \frac{S}{Q}n_j$$

$\bar{x}$	funkce průměrné zásoby
T	jednotka časového období
n_s	náklady na udržování zásob z průměrné zásoby [Kč]
c	cena skladované položky [Kč]
S	stanovená spotřeba
Q	velikost dodávky
n_j	jednorázové náklady na jednu objednávku

A hledaná funkce je tedy: $\min N = N_1 + N_2$

V případě stochastických modelů je navíc třeba brát v úvahu i ztráty (N_3) spojené s nedostatkem surovin či materiálu:

$$N_3 = \bar{x}_z n_z$$

$\bar{x}_z$ ztráty při průměrném chybějícím množství

n_z ztráty způsobené chybějící položkou [Kč]

A hledaná funkce je tedy: $\min N = N_1 + N_2 + N_3$

2.2 Funkce zásob

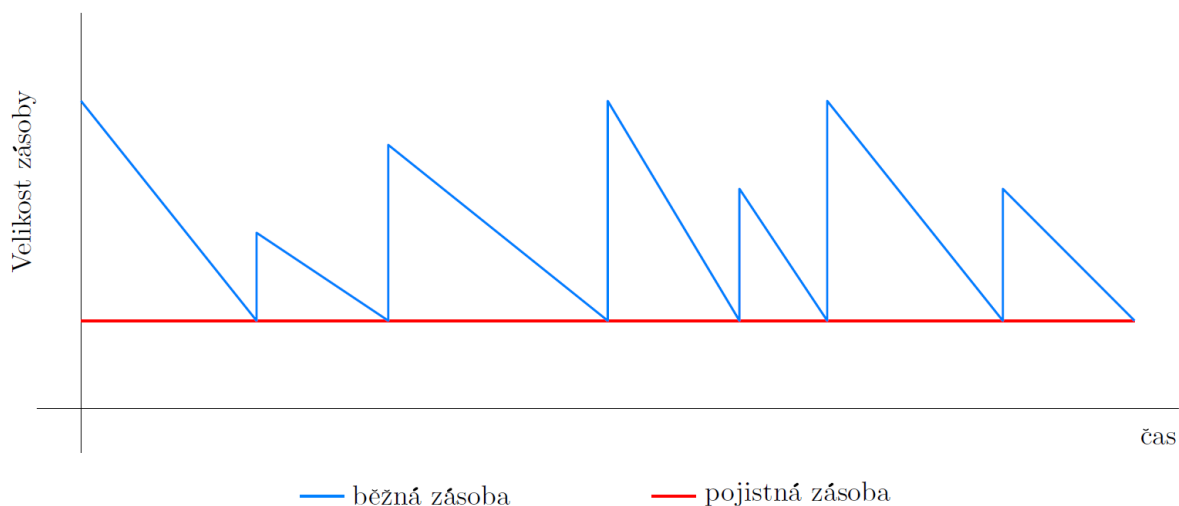
Pro potřeby práce je třeba rozlišovat dva druhy zásob:

Obratová (běžná) zásoba

Jde o hlavní funkci zásob, která má za úkol zabezpečit plynulost výroby. Je proměnná v čase, závislá na způsobu pořizování a čerpání zásob. Vzniká jako důsledek nespojité přepravy od dodavatele k odběrateli.

Pojistná zásoba

Je konstantní v čase, využívá se jako rezerva před náhodnými výkyvy poptávky (potřeby), případně před poruchami či přerušení dodávek. Ve stochastických systémech řízení zásob hraje klíčovou roli právě z důvodu náhodného vzniku nežádoucích jevů, které mohou zpozdit zásobování, případně navýšit poptávku (potřebu).



Graf 1 Příklad běžné a pojistné zásoby

2.3 Druhy nákladů

Stochastické optimalizační procesy se stejně jako procesy deterministické zabývají takovými systémy zásob, u kterých kritériální funkce závisí na následujících druzích nákladů (ztrát):

Náklady na pořízení zásob

Dělí se na fixní a variabilní. Fixní náklady jsou nezávislé na velikosti dodávky, naopak variabilní náklady jsou přímo úměrně závislé na množství zásob. Pro určení velikosti zásob jsou samozřejmě rozhodující náklady variabilní, které jsou přímo úměrné velikosti dodávky. Fixní náklady jsou například náklady administrativní.

Náklady na skladování zásob

Jsou to náklady, které závisí na stavu zásob a době jejich skladování. Složkami v těchto nákladech jsou úroky z oběžných prostředků, které jsou vázány v zásobách, odvody ze zásob, vlastní náklady vynaložené na udržení a na manipulaci zásob včetně administrativních nákladů na evidenci. Dalšími složkami jsou i ztráty v důsledku přirozeného úbytku nebo zastarávání zásob a náklady na pojištění zásob a skladů.

Náklady (ztráty) v důsledku nedostatku zásob

Vznikají, když poptávka není uspokojena v plné výši, nebo je uspokojena opožděně. Pokud neexistují zásoby v době potřeby, je někdy možné je pořídit rychleji, než je běžné, avšak za vyšší cenu. Tím vznikají právě dodatečné náklady. Pokud ovšem nelze v době, kdy je zvýšená poptávka, pořídit zboží ani za vyšší pořizovací cenu, případně by to bylo nevýhodné, vzniká zmařená příležitost, která vede ke ztrátám. Tyto ztráty se zpravidla dají pouze odhadovat, protože nelze přesně určit velikost neuspokojené poptávky. Do nákladů patří také penále.

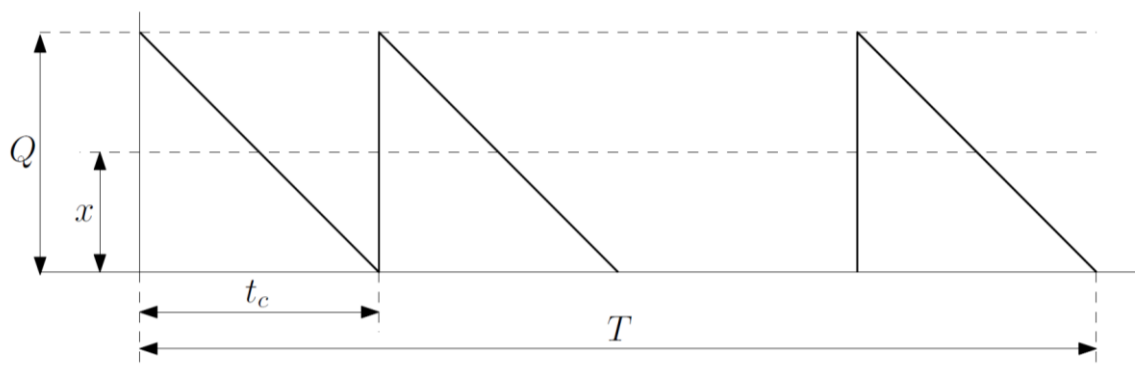
Všechny tyto druhy nákladů spolu navzájem souvisí, kdy snížením jednoho druhu nákladů se zvýší jeden nebo oba dva zbývající druhy. Vhodná strategie tedy umožňuje ovlivnit výši očekávaných celkových nákladů. [1]

3 DETERMINISTICKÉ MODEL Y ŘÍZENÍ ZÁS OB

Deterministické modely jsou použitelné pouze pro ty jednodušší případy, kdy je možné určit požadavky na nákup surovin či polotovarů, nebo je možné vyjádřit poptávku za zvolené časové období jako konstantu.

3.1 Optimální velikost objednávky

Jedná se o nejjednodušší případ deterministických modelů. Předpokládá se, že je poptávka stanovena přesně. Jsou také známy náklady na udržování zásob z průměrné zásoby (n_s) a jednorázové náklady na jednu objednávku (n_j). Předpokládá se lineární spotřeba zásob.



Graf 2 Časový průběh velikosti zásob při modelu optimální velikosti objednávky

Z grafu je zřetelný deterministický vývoj, stejné časové intervaly mezi dodávkami a stejná výše objednávek. V době dodání objednávky na sklad je na skladě předpokládané množství zásob $x_{max} = Q$ a při vyčerpání zásob $x_{min} = 0$. Z toho lze jednoduše odvodit, že průměrná zásoba je rovna polovině velikosti dodávky ($\bar{x} = Q/2$), počet objednávek je podíl stanovené spotřeby proti velikosti dodávky ($o = S/Q$). [2]

Nákladová funkce je tedy následující: $N(Q) = \frac{Q}{2} T n_s c + \frac{S}{Q} n_j$

Po úpravě vychází optimální velikost objednávky: $Q_{opt} = \sqrt{\frac{2S n_j}{T n_s c}}$

Po dosazení Q_{opt} do nákladové funkce je získán vztah pro optimální náklady:

$$N(Q_{opt}) = \sqrt{2STcn_s n_j}$$

Jako poslední je třeba určit dodací cykly, tedy dobu mezi objednávkami (dodávkami na sklad):

$$t_c = \frac{T}{(S/Q_{opt})} = \sqrt{\frac{2T n_j}{S n_s c}}$$

3.1.1 Doplnění o přírážku k ceně

Je běžné, že dodavatel nebude vždy akceptovat požadavky zákazníka. Velikost objednávek může znamenat nižší využití dopravních prostředků nebo může být nevyhovující z hlediska výroby či prodeje (např. malé množství nelze vyrobit apod.). [2]

Je tedy otázkou, jestli se v daném rozmezí přiklonit spíše k menšímu nebo spíše k většímu množství. Z úpravy přírůstkové funkce vyplyne následující výraz:

$$\frac{N(Q)}{N(Q_{opt})} = \frac{1}{2} \left[\frac{Q}{Q_{opt}} + \frac{Q_{opt}}{Q} \right]$$

Z uvedeného výrazu plyne, že růst funkce je pro hodnoty $Q > Q_{opt}$ pomalejší, než pro hodnoty $Q < Q_{opt}$. Obecně lze tedy říci, že je výhodnější přijímat spíše větší objednávky. [2]

3.1.2 Přírážka při různém množství

V jistých případech se stává, že dodavatel požaduje za jiné než obvyklé množství balení nebo dodávky přírážku k ceně. V takovém případě je nutné vypočítat celkové náklady na množství menší než optimální a větší než optimální při běžných cenách a dále pak vypočítat za cenu s přírážkou celkové náklady na optimální množství. Z výsledku je pak možné rozhodnout, k jaké variantě se přiklonit. [2]

3.1.3 Doba potřebná k vyřízení objednávky

Pokud nastane případ, že termín pro vyřízení objednávky není nulový ($t_{vo} > 0$), je třeba určit dolní objednávací mez (signální stav zásob) x_d . Jakmile zásoba klesne pod tento stav, je třeba vystavit objednávku na běžně stanovené množství zásob. Signální stav zásob se rovná $x_d = st_{vo}$, přičemž $s = S/T$ a jedná se o průměrnou denní spotřebu požadované položky. Dále mohou nastat následující dva případy. Buď je $x_d = Q$, tedy $t_{vo} = t_c$ a je třeba objednat novou dávku hned při doručení objednávky minulé, nebo dojde k tomu, že $t_{vo} > t_c$ a není tedy možné dosáhnout takového stavu zásob. V tomto případě se signální stav zásob určuje z neceločíselné části $s \cdot t_{vo}/Q$ a celočíselná část říká, kolik přijde na sklad dodávek během doby T . [2]

3.2 Optimální velikost objednávky při nespojitém množství

V logistických řetězcích je poměrně běžné, že se výrobky balí po více kusech a je možné je koupit pouze v tomto balení. Z tohoto důvodu je třeba rozhodnout, kolik balení koupit, pokud se počet kusů liší od optimálního množství.

Pokud je balení q , potom je možné objednat pouze v násobcích, tedy $Q = q, 2q, 3q, \dots$ a musí platit následující nerovnost:

$$N(Q_{opt} + q) \geq N(Q_{opt}) \geq N(Q_{opt} - q)$$

Po doplnění a úpravě:

$$Q(Q + q) \geq \frac{2Sn_j}{Tcn_s} \geq Q(Q - q)$$

Optimální velikost objednávky je možné nalézt způsobem, kdy je třeba nejprve vypočítat podíl $2Sn_j/Tcn_s$ a následně se sestaví tabulka, do které se budou dopočítávat pravé a levé členy nerovnosti, dokud nebude pro příslušné Q nerovnost splněna. [2]

3.3 Optimální velikost objednávky při množstevních rabatech

Jeden z dalších, běžně užívaných způsobů z nabídky dodavatelů, je nabízet množstevní slevy (rabaty). S rostoucí velikostí objednávky klesá cena. V některých případech jde o funkci, kdy každý kus sníží cenu, ovšem ve většině případů se využívá intervalů, kdy je cena v jednotlivých intervalech vždy stejná, viz následující tabulka. [2]

Tab. 1 Cena v jednotlivých intervalech

Interval množství	Cena za množstevní jednotku
$Q_0 - Q_1$	c_1
$Q_1 - Q_2$	c_2
$Q_2 - Q_3$	c_2
...	...
$Q_{n-1} - Q_n$	c_n

V tomto případě je možné uplatňovat 2 druhy slev:

3.3.1 Sleva A

Při tomto způsobu je sleva uplatňována na celou objednávku podle toho, do jakého intervalu spadá celkové množství.

Za dodané množství S v období T je fakturována částka:

$$F = c_i S$$

c_i cena připadající k intervalu, do kterého spadá množství Q

Při volbě p -tého intervalu ($n_p + c_p Q$) pro výpočet celkových nákladů lze použít následující vzorec:

$$N(Q) = 0,5(Tn_s n_j + Tn_s c_p Q) + (S/Q)n_j + Sc_p$$

Optimální velikost objednávky v i -tém je možné získat pomocí vztahu při derivaci Q rovné nule:

$$Q_{p,opt} = \sqrt{\frac{2Sn_j}{Tc_p n_s}}$$

Výsledná nákladová funkce křivka pro interval (Q_{p-1}, Q_p) :

$$N_p(Q_{p,opt}) = 0,5Tn_s n_j + Sc_p + \sqrt{2TSc_p n_s n_j}$$

Postup pro hledání optimální velikosti objednávky je následující:

Pro poslední, n -tý interval se využije nejnižší cenu c_n pro výpočet optimální velikosti objednávky podle vzorce $Q_{k,opt} = \sqrt{2Sn_j/Tc_p n_s}$. Pokud vypočtená velikost leží v intervalu (Q_{k-1}, Q_k) , je vypočtená velikost skutečně optimální. Jestliže tomu tak není, vypočítá se $Q_{n-1,opt}$ při ceně c_{n-1} a znovu se ověří, zda je množství v intervalu (Q_{k-2}, Q_{k-1}) . Tento postup se aplikuje do té doby, dokud vypočtené množství nespadá do příslušného intervalu. [2]

Ve chvíli, kdy vypočtené množství spadá do příslušného intervalu, následuje vypočtení příslušných minimálních nákladů, kde spodní index n značí posun horní hranice intervalu:

$$N_{k-n}(Q_{k-n,opt}) = 0,5Tn_s n_j + Sc_{k-n} + \sqrt{2TSc_{k-n} n_s n_j}$$

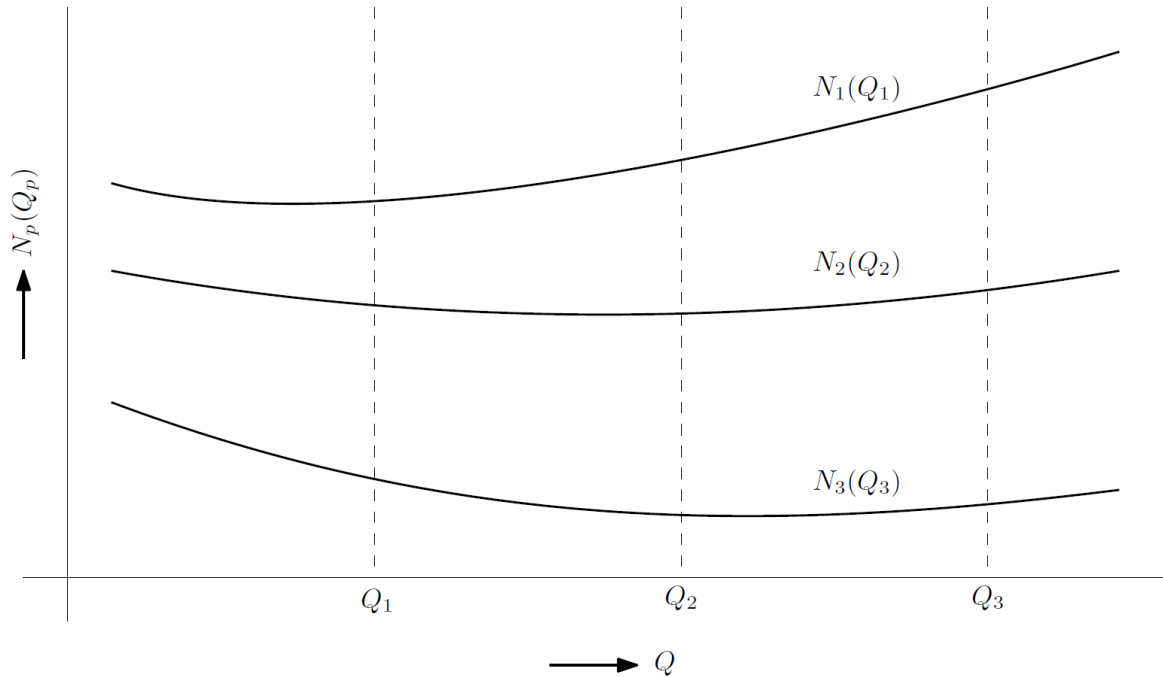
A pro případ, kdy by místo optimální množství byla použita horní hranice intervalu, je třeba vypočítat také tyto náklady:

$$N_k(Q_{k-(n+1)}) = 0,5Tn_s n_j + Sc_{k-(n+1)} + \sqrt{2TSc_{k-(n+1)} n_s n_j}$$

Bude-platit následující nerovnost, bude $Q_{k-1,opt}$ optimální velikostí objednávky:

$$N_{k-n}(Q_{k-n,opt}) < N_k(Q_{k-1})$$

Pokud tomu tak nebude, je výhodnější za optimum zvolit dolní mez k -tého intervalu, tedy Q_{k-n} .



Graf 3 Velikost nákladových křivek vzhledem k obj. množství u slevy typu A

3.3.2 Sleva B

V tomto případě je vždy fakturována cena v daném intervalu za množství, které tento interval pokrývá. Za každý interval se vypočítá cena podle množství a ceny se sečtou, viz následující příklad pro tři intervaly a objednávané množství Q :

$$F = (Q_1 - Q_0)c_1 + (Q_2 - Q_1)c_2 + (Q - Q_2)c_3$$

Pro p -tý interval je částka za objednávku rovna:

$$F_p = (Q - Q_{p-1})c_p + \sum_{s=1}^{p-1} (Q_s - Q_{s-1})c_s$$

Přičemž průměrná cena za dodanou jednotku je:

$$\bar{c} = \frac{F_p}{Q} = \frac{Q - Q_p}{Q}c_p + \frac{F_{p-1}}{Q}$$

$$\text{kde } F_{p-1} = \sum_{s=1}^{p-1} (Q_s - Q_{s-1})c_s$$

Nákladová křivka vychází z následující vzorce:

$$N_p(Q) = (S/Q)(n_j + F_{p-1} - Q_{p-1}c_p) + Sc_p + 0,5Tn_s(Qc_p - Q_{p-1} + F_{p-1})$$

Pro výpočet optimální velikosti objednávky v p -tém intervalu poslouží derivace podle Q rovna nule:

$$Q_{p,opt} = \sqrt{\frac{2S(n_j + F_{p-1} - Q_{p-1}c_p)}{Tc_p n_s}}$$

Jako poslední zbývá vypočítat optimální náklady, které se získají dosazením optimálního množství do vztahu pro $N_p(Q)$:

$$N_p(Q_{p,opt}) = SC_p + 0,5Tn_s(F_{p-1} - Q_{p-1}c_p) + \sqrt{2STn_s(n_j + F_{p-1} - Q_{p-1}c_p)c_i}$$

Postup pro hledání optimální velikosti objednávky je jednodušší než v předchozím případě. Pro každý interval hodnot je třeba vypočítat optimální objednávkové množství $Q_{p,opt}$. Pokud ne, přechází se k dalšímu intervalu. Pokud ano, vypočte se nákladová funkce. Z použitelných intervalů se vybírá ten s nejnižšími náklady. [2]

4 STOCHASTICKÉ MODEL Y ŘÍZENÍ ZÁS OB

Úkolem stochastických modelů řízení zásob je odpovědět na otázky, které jsou důležité pro rozhodování a co nejvýhodnější fungování skladu. Konkrétně to jsou otázky: Kolik zásob objednat? Kdy zásoby objednat? Jakou velikost pojistné zásoby vytvořit? Předpokladem je zde právě náhodná velikost poptávky, jejíž míru při výpočtech vyjadřujeme pravděpodobnostním rozložením.

4.1 Charakteristiky systémů zásob

Pro správný popis systému zásob je nejprve třeba uvést jeho charakteristiky, které se dělí podle toho, zda je můžeme měnit, na říditelné a pokud nemůžeme, tak na neříditelné. Říditelné proměnné pomáhají určit, kdy vytvářet či doplňovat zásoby a v jaké výši, přičemž mohou být nezávislé nebo se naopak vzájemně ovlivňovat. [2]

4.1.1 Říditelné proměnné

Objem zdrojů požadovaných na sklad

Objem zdrojů se nejčastěji řídí dvěma způsoby. První způsob je takový, že se zásoby vytvářejí a doplňují v dávkách neměnné velikosti. V druhém případě se vytvářejí a doplňují v dávkách různé velikosti. Navíc existují i systémy zásob, které regulaci vytváření či doplňování zásob zcela neumožňují, protože objem nebo intenzita dodávky je určena pouze pravděpodobnostním způsobem. [2]

Při tvorbě zásob se rozlišuje na zásobu obratovou (provozní) a na zásobu pojistnou. Obratová zásoba je vytvářena z objektivních důvodů, kdy dochází k časovému posunu mezi výrobou nebo naskladněním. Pojistná zásoba je tvořena pro pokrytí výkyvů, ke kterým může dojít v případě výpadku výrobního či dopravního řetězce, anebo pro pokrytí zvýšené poptávky. [2]

Frekvence nebo termín vyžádání zdrojů na sklad

Dalším faktorem, který je třeba určit při vytváření systémů zásob je intenzita, s jakou se zásoby dostávají na sklad. Opět rozlišujeme dva způsoby řízení, kdy jedním z nich je pevně daná frekvence dodávek a druhým z nich je proměnná frekvence dodávek. Dobu mezi dvěma následujícími objednávkami nazýváme dodací nebo objednací lhůtou, nebo dodacím či objednacím cyklem. Stejně jako v případě řízení objemu je zde navíc případ, kdy nelze frekvenci či dobu dodání na sklad přesně určit, protože mají stochastický charakter. [2]

V praxi je možné se setkat se systémy zásob, které umožňují řídit obě dvě proměnné nebo pouze jednu z nich, tedy množství naskladněné zásoby nebo frekvenci.

4.1.2 Neříditelné proměnné

Velikost potřeby (poptávky)

Během určitého časového intervalu vyvstává velikost potřeby ve výrobních procesech, případně velikost poptávky v zákaznickém sektoru. Tuto proměnnou nelze přímo a obvykle ani nepřím ovlivňovat. Právě v této proměnné se nejvíce projevuje rozdíl mezi deterministickým a stochastickým systémem. Zatímco systém deterministický má velikost potřeby či poptávky

v jednotlivých intervalech přesně známou, stochastický systém tuto informaci neobsahuje a bývá v něm nahrazována pravděpodobnostním rozložením nebo pomocí empirického rozložení (vycházející z pozorování). Je třeba dále brát v úvahu to, že jednotlivá rozhodnutí o výši či okamžiku doplnění se mohou vzájemně ovlivňovat.

Dále se pak podle potřeby (poptávky) dělíme úlohu na statickou a dynamickou. Statickou úlohu neovlivňují sezónní vlivy ani jiné výkyvy. Je tedy konstantní. Dynamická úloha pak zahrnuje vlivy, které ovlivňují velikost potřeby (poptávky). Protože již její velikost není konstantní a nelze ani přesně určit, jaká bude její velikost, nahrazujeme její velikost pravděpodobnostním rozložením, které je vestaveno s přihlédnutím na předchozí zkušenosti případně pak na expertní úsudek. Na tento druh úlohy nelze vždy aplikovat analytické metody a jsou pak řešeny aproximačními způsoby numerického řešení, často pak simulačními metodami.

Pořizovací lhůta zásob

Doba, která uběhne mezi vznikem objednávky a příchodem zásob na sklad. Jen v málo případech lze tuto dobu ovlivnit, většinou je pevně daná. V některých případech je pořizovací lhůta zanedbatelně malá a proto ji není třeba brát v úvahu. Stejně jako v případě potřeby (poptávky) je i pořizovací lhůta zásob pevně daná nebo proměnlivá, tedy má buď deterministický, nebo stochastický charakter.

Právě charakter neřiditelných proměnných, velikosti potřeby (poptávky) a pořizovací lhůta zásob, je jedním z hlavních kritérií klasifikace.

Zatímco u deterministických systému řízení zásob je cílem určit velikost obrátové (provozní) zásoby, je hlavním cílem optimalizace ve stochastickém systému řízení zásob je stanovení pojistné zásoby, která bude s požadovanou pravděpodobností zaručovat minimum očekávaných celkových nákladů.

4.2 Klasifikace stochastických systémů řízení zásob

Obdobně jako se u deterministických systémů řízení zásob, které mají buď konstantní poptávku (potřebu), příp. pořizovací lhůtu nebo ji mají v čase proměnnou, stochastické systémy mají pravděpodobnostní rozdělení konstantní nebo se s časem mění, přičemž tuto změnu lze většinou pouze aproximativně odhadnout. Pokud jsou poptávka (potřeba), příp. pořizovací lhůta náhodnými veličinami, je náhodný i stav zásob v okamžiku dodání zásoby na sklad. Právě z této úvahy vyplývá, že hlavní úlohou stochastických systémů je stanovení výše pojistné zásoby. Ta je potřebná pro plynulý chod provozu. Nejčastěji používaná teoretická rozdělení jsou normální, exponenciální nebo Poissonovo rozdělení, případně rozdělení empirická. [2]

4.2.1 Jednorázová objednávka

Nejjednodušší případ stochastického modelu řízení zásob je situace, kde je potřeba vytvořit jednorázovou zásobu na pokrytí poptávky v určitém období T . Pokud je možné každému náhodnému výskytu velikosti skutečné poptávky S přiřadit pravděpodobnost výskytu $P(S)$, tak pro dané období T je objednáno celkem Q jednotek požadovaného zboží. Po uběhnutí určeného období mohou nastat dvě krajní situace, a to:

$S \leq Q$, na skladě zůstane nevyužito množství $Q - S$

$S \leq Q$, na skladě bude chybět $S - Q$ jednotek

Pro následující výpočet jsou použity náklady na pořízení jednotky nakupovaného zboží c a vícenásobné náklady na dokoupení jednotky chybějícího množství či přímo ztráty z jeho nedostatku c_z . Pro zvolenou strategii není třeba brát v úvahu náklady na udržování zásob. Samotný výpočet výše nákladů nebo ztrát je následující:

$$N(Q) = \sum_{S=0}^Q (Q - S)P(S)c + \sum_{S=Q+1}^{\infty} (S - Q)P(S)c_z$$

Po úpravě odvozených výrazů pro $N(Q + 1)$ a $N(Q - 1)$ získáme nerovnost

$$P(S \leq (Q - 1)) \leq \frac{c_z}{c_z + c} \leq P(S \leq (Q))$$

kde použité výrazy jsou rovny

$$P(S \leq (Q - 1)) = \sum_{i=0}^{Q-1} P(S)$$

$$P(S \leq (Q)) = \sum_{i=0}^Q P(S)$$

Prakticky to tedy znamená, že pro nalezení optimálního množství Q_{opt} stačí vypočítat hodnotu $c_z/(c_z + c)$ a nalézt nejvhodnější hodnotu distribuční funkce, která bude pro danou podmínku splněna.

4.2.2 Modely řízení zásob s náhodnou a proměnlivou spotřebou v čase

Je třeba, aby systém řízení zásob respektoval spolu s náhodnými změnami poptávky také fakt, že spotřeba zásoby probíhá v čase nerovnoměrně. Expedice, ať už surovin, polotovarů nebo výrobků, je uskutečňována po určitých jednorázových dávkách a jediný případ, kdy lze pokles vyjádřit přímkou, je kontinuální výroba. V případech nerovnoměrné spotřeby je třeba prozatím jednorázová rozhodnutí nahradit rozhodováním, které bude korigovat naše předcházející chybná rozhodnutí a umožní reagovat podle měnící se situace. V podmínkách spotřeby proměnné v čase je třeba opět odpovědět na dvě základní otázky, které už byly vysloveny dříve:

Kdy zásoby doplňovat?

V jak velkých dávkách doplňovat?

Odpověď na první otázku, tedy kdy zásoby doplňovat, lze najít pomocí následujících dvou možností:

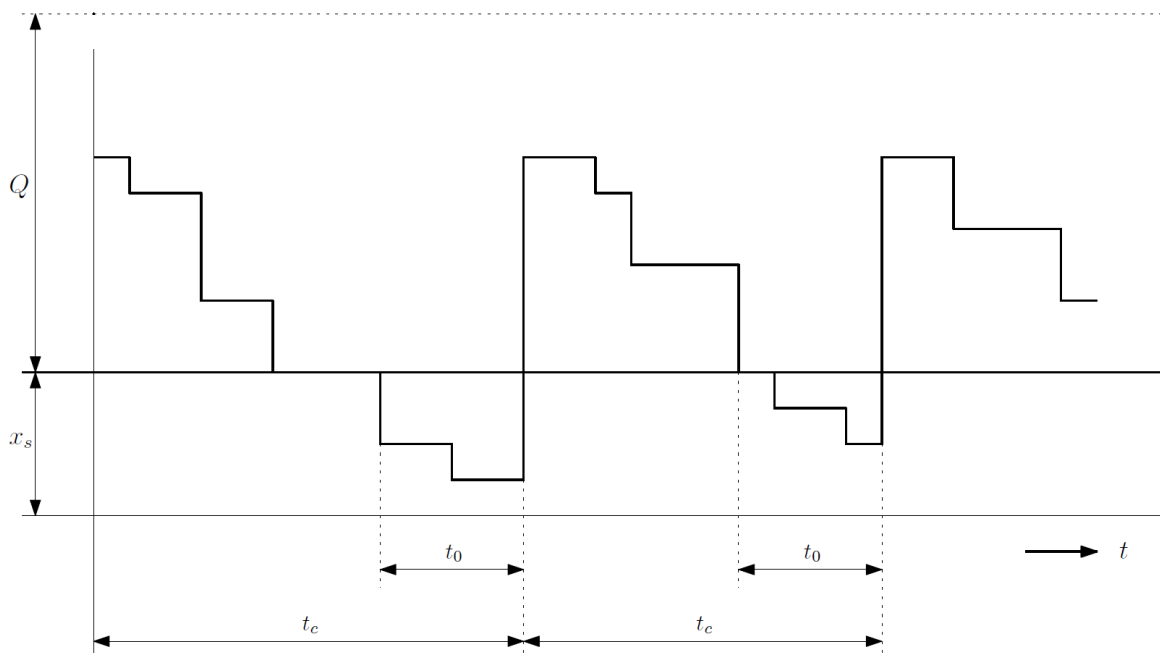
1. Na ose y , která reprezentuje množství zásoby, určit spodní mez x_z , nazývanou spodní objednávací úroveň nebo také signální stav zásob. Ve chvíli, kdy skutečný stav zásob

klesne pod hodnotu této meze, je to signál pro doplnění zásob. Tyto systémy jsou označovány jako **Q-systémy**. Použité písmeno Q je zde z toho důvodu, že zásoba je ve většině případů doplňována konstantní velikostí objednávky Q .

2. Osu x , která vyjadřuje čas, je možné rozdělit na stejné intervaly ΔT , vyjadřující určité konstantní časové období, např. týden či měsíc. Určí se okamžik, ve kterém se bude zjišťovat skutečný stav zásoby x , např. pondělí či první pracovní den v měsíci, a na základě tohoto zjištění se bude vždy objednávat množství $Q = x_h - x$, kde x_h je označení pro horní objednávací mez. Tyto systémy jsou známy jako **P-systémy**.

4.2.2.1 Q-systém

Z grafu č. 2 je zřejmá funkce tohoto systému. Zásoby jsou objednávány v různých časových okamžicích, vždy ve stejném množství.



Graf 4 Q-systém

Kritický časový úsek pro funkci Q-systému je termín vyřízení objednávky t_o , zvaný interval nejistoty. V tomto okamžiku může dojít k předčasnému vyčerpání zásoby, což by vedlo ke ztrátám. Aby se dařilo tomuto stavu vyhýbat, je třeba, aby byla signální zásoba schopna pokrýt jednak poptávku po t_o časových jednotkách, tak i možné nepředvídatelné výkyvy v poptávce.

Ke splnění takového úkolu je třeba určit rozdělení pravděpodobnosti $f(s)$ náhodné poptávky s po dobu t_o , a také odhadnout její střední hodnotu a směrodatnou odchylku. Obvykle využívaným zdrojem pro odhad pravděpodobnosti jsou údaje o předchozí poptávce s_i za dostatečně dlouhé uplynulé období T . Jako výhodné řešení se nabízí roztrždit data o poptávce podle intervalů, např. po dnech do tabulky rozdělené podle četností a z ní pak určit potřebné hodnoty. [2]

Střední hodnota:
$$\bar{s} = \frac{\sum_{i=1}^n s'_i n_i}{n}$$

Směrodatná odchylka:
$$\sigma_s = \sqrt{\sum_{i=1}^n (s'_i - \bar{s})^2 n_i / n}$$

(n_i jsou četnosti hodnot v jednotlivých intervalech a s'_i středy intervalů)

Aby bylo možné pokrýt náhodné výkyvy v poptávce, je také nutné určit výši pojistné zásoby x_p . Je třeba zjistit, jestli náhodná poptávka vyhovuje normálnímu pravděpodobnostnímu rozdělení a také vyrovnat empirické rozdělení pravděpodobnosti, které jsme získali analýzou poptávky za minulé období. Postup je následující:

- Vypočítat normované proměnné normální rozdělení pravděpodobnosti pro horní hranice intervalů podle vzorce $u_i = \frac{s_i^{(H)} - \bar{s}}{\sigma_i}$.
- Nalézt hodnoty distribuční funkce normálního rozdělení pravděpodobnosti $F(u_i)$ a vypočítat jednotlivé difference $\Delta_i = F_{i+1}(u_{i+1}) - F_i(u_i)$.
- Vypočítat teoretické četnosti $n_i^{(T)} = n \Delta_i$.
- Vypočítat hodnoty rozdělení $\chi^2 = \sum_{i=1}^e \frac{n_i^2}{n_i^{(T)}} - n$. Podmínka pro použití uvedeného vztahu je, že součty teoretických a empirických četností jsou si rovny.
- Ve statistických tabulkách kvantilů lze nalézt rozdělení chí kvadrát pro zvolenou hladinu významnosti (pro běžnou praxi postačuje pro 95 %) a počet stupňů volnosti $c - 3$ (kde c je počet intervalů) příslušný kvantil q . Pokud pro vypočtené hodnoty platí, že $\chi^2 \leq q$, potom není možné předpoklad normality rozdělení zamítnout. Pokud poptávka vyhovuje normálnímu rozdělení pravděpodobnosti, je možné odhadnout výši pojistné zásoby, zatím bez ohledu na náklady pro udržování dané zásoby, na velikost $x_p = k \sigma_i$.

Pokud je tedy možné určit typ rozdělení pravděpodobnosti poptávky, lze získat sumu nákladů spojených s pořizováním a skladováním zásob a případných ztrát spojených s nedostatkem zásob na skladě jako funkci pojistné zásoby a průměrné doby dodacího cyklu. Aby bylo možné formulovat účelovou funkci, je nutné přijmout následná zjednodušení:

Pořizovací náklady:
$$N_1 = n_j T / \bar{t}_c$$

Náklady na udržování běžné zásoby na skladě:
$$N_2 = 0,5 n_s c \bar{t}_c S$$

Náklady na udržování pojistné zásoby:
$$N'_1 = n_s c x_p T$$

Ztráta z příp. nedostatku zásob na skladě:
$$N_3 = \frac{T n_z}{\bar{t}_c} \int_{x_z}^{\infty} f(s) ds \quad , \quad \text{kde } x_z = x_p + t_0 \bar{s}$$

Výsledná hodnota integrálu představuje pravděpodobnost, s jakou dojde k situaci, kdy bude potřeba větší než stanovená signální zásoba. Odhad ztrát z nedostatku zásob je označen n_z . Počet dodacích cyklů, ve kterých může nastat situace, že dojde k nedostatku zásob je označen podílem $T / \bar{t}_c$.

Závislost celkových nákladů a ztrát na hledané velikosti pojistné zásoby a dodacího cyklu je vyjádřena následující funkcí:

$$N(x_p, \bar{t}_c) = n_f T / \bar{t}_c + 0,5 n_s c \bar{t}_c S + n_s c x_p T + \frac{T n_z}{\bar{t}_c} \int_{x_z}^{\infty} f(s) ds$$

Derivaci funkce podle $\bar{t}_c$ je nutno položit rovnu nule a výsledkem je výraz pro $\bar{t}_c$:

$$t_c^2 = \frac{2T(n_f + n_z(1 - F(x_z)))}{S c n_z^2}$$

Derivaci funkce podle x_p je opět nutno položit rovnu nule a výsledkem pro x_p je výraz:

$$t_c = \frac{n_z f(x_z)}{c n_z}$$

Vhodnou úpravou a dosazením za $x_z = x_p + t_0 \bar{s}$ je výsledkem následující rovnice:

$$f^2(x_p + t_0 \bar{s}) = \frac{2T c n_z (n_f + n_z(1 - F(x_p + t_0 \bar{s})))}{S n_z^2}$$

Z této funkce je možné vhodnou metodou určit výši pojistné zásoby a to s ohledem na nákladová kritéria. Pro tuto metodu jsou využity normované proměnné normálního rozdělení, kdy stačí do vztahu dosadit pro normovanou proměnnou výraz $u = (s - \bar{s})/\sigma_z$ za hodnotu s , pro kterou potřebujeme znát hustotu pravděpodobností f a distribuční funkci F , tedy:

$$s = x_z = x_p + t_0 \bar{s} = k \sigma_z + t_0 \bar{s}$$

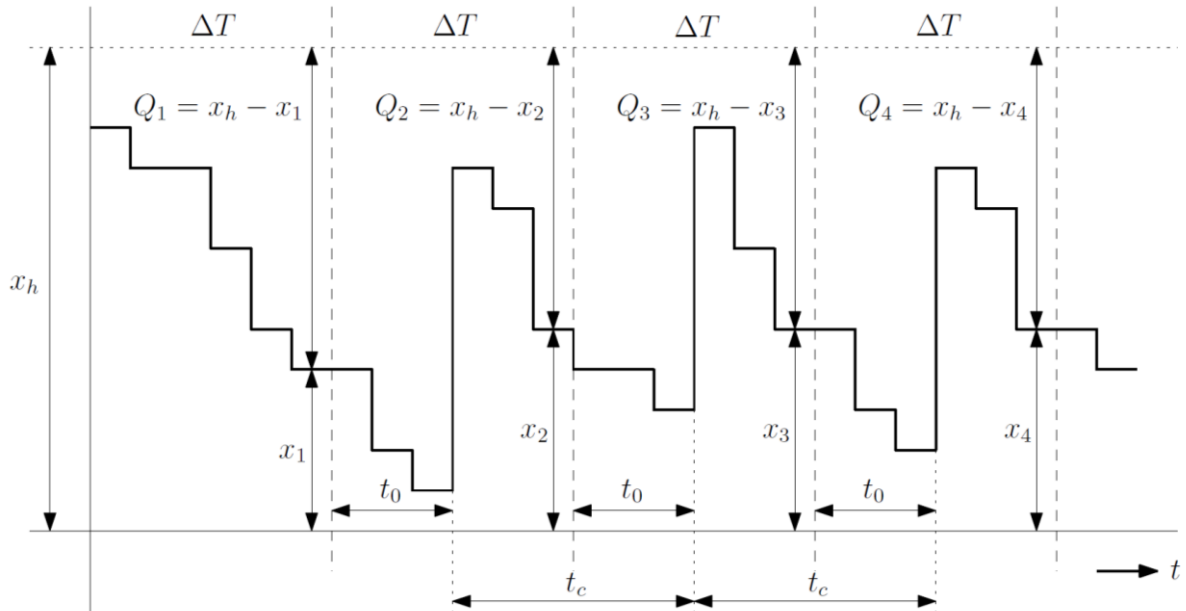
Za $\bar{s}$ se dosadí průměrná spotřeba a výsledkem pro $t_0 = 1$ je výraz:

$$2\varphi^2(k) = \frac{2T c n_z (n_f + n_z(1 - \Phi(k)))}{S n_z^2} \sigma_z^2 = A$$

Je to z důvodu platnosti vztahu $f(s) = \varphi(k)/\sigma_s$ mezi hustotami pravděpodobností pro původní hodnoty f a normované proměnné φ . Postupné dosazování vede k nalezení hodnoty k . A díky k je možné vypočítat hodnotu pojistné zásoby ze vzorce $x_p = k \sigma_s$ a dále nalézt délku dodacího cyklu t_c . Z něho je poté možné zjistit počet objednávek podle vzorce $o = T/t_c$ a následně velikost objednávek $Q = S/o$. [2]

4.2.2.2 P-systém

P-systém je opakem Q-systému. Zatímco v předchozím případě je pevná velikost objednávky a proměnné časové okamžiky objednání, v případě P-systému se ve stejných časových okamžicích objednávají různá množství podle $Q = x - x_h$, kde x_h je stav zásob v okamžiku objednávky.



Graf 5 P-systém

Protože jsou zde pevně stanovené okamžiky objednání, je interval nejistoty delší než u předchozího systému. Pro příklad, pokud by se neobjednalo v jednom termínu dostatečné množství, je třeba čekat do termínu dalšího na doobjednání. Aby byla vytvořena dostatečná zásoba, je třeba, aby interval nejistoty byl roven součtu průměrného termínu vyřízení objednávky a průměrného dodacího cyklu. Horní objednávací úroveň je tedy vyjádřena vztahem $x_h = x_p + (t_o + t_c)$. Při formulaci nákladové funkce umožňující určit řídicí veličiny systému vyvstává problém, který spočívá v neznalosti průměrné délky dodacího cyklu, která je potřebná pro charakteristiku rozdělení pravděpodobností v celém intervalu nejistoty. Ovšem pro vybraná rozdělení pravděpodobností lze nalézt průměrnou spotřebu $\bar{s}$ a směrodatnou odchylku σ_s pro předem stanovené období, tedy např. týden. Potom lze použít zjednodušení, kdy spotřeba v k -krát delším období bude rovna $k\bar{s}$ a směrodatná odchylka bude rovna $\sigma_s\sqrt{k}$. Hustota pravděpodobností je pak vyjádřena násobností exponentu $f^k(s)$. Tyto vztahy lze použít např. pro rozdělení normální, binomické, Poissonovo apod. Pro poptávku řízenou normálním rozdělením bude nákladová funkce vyjádřena následujícím vztahem:

$$N(x_p, \bar{t}_c) = n_j T / \bar{t}_c + 0,5 n_s c \bar{t}_c S + n_s c x_p T + \frac{T n_z}{\bar{t}_c} \int_{x_h}^{\infty} f^{(t_o + \bar{t}_c)}(s) ds$$

Je možné využít také další postup, který je podobný jako u Q-systému. Protože je zde problém s odhadem rozdělení pravděpodobností, využívá se jednoduchý aproximativní postup, který je možné použít i u Q-systému:

Odhadne se průměrná velikost objednávky a počet objednávek z následujících vztahů:

$$\bar{Q}_{opt} = \sqrt{\frac{2Sn_j}{Tcn_s}} \quad \bar{t}_{opt} = \frac{T}{(S/Q_{opt})} \quad o_{opt} = \frac{T}{t_c}$$

Průměrná velikost objednávky (jedná se o P-systém, velikost objednávky se bude v průběhu měnit) se vyjádří ze vztahu:

$$\bar{Q}_{opt} = (t_0 + \bar{t}_c)\bar{s}$$

Náklady jsou pak formulovány jako funkce pojistné zásoby:

$$N(x_p) = x_p cn_s T + o_{opt} n_z \int_{x_p + \bar{Q}}^{\infty} f^{(t_0 + \bar{t}_c)}(s) ds$$

Z derivace podle x_p rovné nule vyplývá:

$$f^{(t_0 + \bar{t}_c)}(x_p + \bar{Q}) = \frac{cn_s T}{o_{opt} n_z} = A$$

Následně po úpravě:

$$\varphi(k) = \frac{cn_s T}{o_{opt} n_z} \sigma_s$$

Po dosazení známých veličin je nalezeno k a z něho následně vypočítána pojistná zásoba a horní objednávací úroveň. [2]

5 PŘÍKLADY

Aby byla možná implementace programu, je třeba také vyjít z příkladů, které poslouží pro lepší pochopení dané problematiky. Příklady týkající se deterministických modelů zde uvedených jsou členěny podle [2]. Q-systém je standardní model, který je popsán v [3]. Výpočet příkladu pro P-systém je převzat z [4].

5.1 Příklad s optimální velikostí objednávky

Pro zabezpečení nepřetržité činnosti kotle je třeba zabezpečit dodávky 1 000 tun černého uhlí. Pořizovací cena včetně dopravy je 6 500 Kč za tunu. Náklady na vyřízení a převzetí dodávky jsou 7 500 Kč. Roční náklady na udržování zásob jsou přibližně 18% průměrné hodnoty zásob za rok. Je potřeba určit, v jak velkých dodávkách a jak často je třeba zásobovat kotel, aby byly náklady co nejnižší.

Tab. 2 Vstupní data příkladu s optimální velikostí objednávky

Údaje	Označení	Hodnoty	Jednotky
Poptávka za období T	S	1 000	Tuny
Náklady na vyřízení objednávky	n_j	7 500	Kč
Procentuální náklady na skladování	n_s	18	% z průměrné zásoby za rok
Cena za jednotku	c	6 500	Kč/tuna
Období	T	1	Rok

Postup výpočtu:

$$Q_{opt} = \sqrt{\frac{2Sn_j}{Tn_sc}} = \sqrt{\frac{2 \cdot 1\,000 \cdot 7\,500}{1 \cdot 0,18 \cdot 6\,500}} = 114 \text{ tun}$$

$$N(Q_{opt}) = \sqrt{2STcn_sn_j} = \sqrt{2 \cdot 1\,000 \cdot 1 \cdot 6\,500 \cdot 0,18 \cdot 7\,500} = 133\,000 \text{ Kč}$$

$$t_c = \sqrt{\frac{2Tn_j}{Sn_sc}} = \sqrt{\frac{2 \cdot 1 \cdot 7\,500}{1\,000 \cdot 0,18 \cdot 6\,500}} \doteq 0,1132 \text{ roku} \doteq 41 \text{ dní}$$

Tab. 3 Výsledky příkladu s optimální velikostí objednávky

Údaje	Označení	Hodnoty	Jednotky
Optimální objednávací množství	Q_{opt}	117	tuny
Náklady při ideálním množství	$N(Q_{opt})$	133 000	Kč
Perioda objednání	t_c	41	dny

5.2 Příklad s optimální velikostí objednávky při nespojitém množství

Pro příklad budou využity stejné hodnoty jako v předchozím příkladu. Ovšem dojde k doplnění zadání, kdy dodavatel nabízí dodávky po 100 tunách za nižší cenu, 6 200 Kč. Pokud bude množství rozdílné, bude cena o 300 Kč vyšší. Pro potřeby příkladu je třeba vypočíst množství 300 tun a 400 tun.

$$N(100) = \frac{Q}{2} T n_s c + \frac{S}{Q} n_j = \frac{100}{2} \cdot 1 \cdot 0,18 \cdot 6\,200 + \frac{1\,000}{100} \cdot 7\,500 \doteq 131\,000 \text{ Kč}$$

$$N(200) = \frac{Q}{2} T n_s c + \frac{S}{Q} n_j = \frac{200}{2} \cdot 1 \cdot 0,18 \cdot 6\,200 + \frac{1\,000}{200} \cdot 7\,500 \doteq 149\,000 \text{ Kč}$$

Je třeba vypočíst také náklady pro optimální velikost objednávky za vyšší cenu:

$$N(114) = \frac{Q}{2} T n_s c + \frac{S}{Q} n_j = \frac{114}{2} \cdot 1 \cdot 0,18 \cdot 6\,500 + \frac{1\,000}{114} \cdot 7\,500 \doteq 133\,000 \text{ Kč}$$

Tab. 4 Vstupní data příkladu s optimální velikostí objednávky při nespojitém množství

Údaje	Označení	Hodnoty	Jednotky
Náklady při 100 kusech	$N(100)$	131 000	Kč
Náklady při 114 kusech	$N(114)$	133 000	Kč
Náklady při 200 kusech	$N(200)$	149 000	Kč

I když se jednotlivé varianty výrazně neliší, je v tomto případě výhodnější přijmout dodávky ve výši 400 tun s dodacím cyklem $(365 \cdot 100)/1\,000 = 36$ dní.

Tab. 5 Výsledky příkladu s optimální velikostí objednávky při nespojitém množství

Údaje	Označení	Hodnoty	Jednotky
Optimální objednávací množství	Q_{opt}	100	tuny
Náklady při určeném množství	$N(Q_{opt})$	131 000	Kč
Perioda objednání	t_c	36	dny

5.3 Příklad s dobou potřebnou pro vyřízení objednávky

Opět budou využité hodnoty z předchozího příkladu. Termín vyřízení objednávky je stanoven na 10 dní. Z tohoto lze vypočítat signální stav zásob:

$$x_d = s \cdot t_{vo} = 1\,000 \cdot \frac{10}{365} = 27 \text{ tun}$$

Pokud je termín vyřízení objednávky větší než perioda objednání, je třeba signální stav zásob upravit a přitom je nutné o násobky, o které převyšuje objednávací množství celou objednávku, násobně vyřídít objednání adekvátní termínu doručení na sklad, protože by jinak nebylo možné pokrýt poptávku.

Pokud je tedy termín objednávky stanoven například na 30 dní, poté je výpočet signálního stavu zásob následující:

$$x_d = s \cdot t_{vo} = 1\,000 \cdot \frac{90}{365} = 246 \text{ tun}$$

$$x_d \rightarrow x_d \bmod Q_{opt} = 18 \text{ tun}$$

Navíc je třeba tuto dobu pokrýt 2 objednávkami navíc.

5.4 Příklad s optimální velikostí objednávky při nespojitém objednacím množství

Je možné objednávat pouze balení po 250 ks výrobků. Potřebné je nalézt optimální velikost objednávek pro další období za předpokladu, že náklady spojené s jednou objednávkou jsou ve výši 10 500 Kč a náklady na udržování zásob jsou 17% z hodnoty průměrné roční zásoby. Cena výrobku je pak 3 200 Kč/kus a očekávaná poptávka 20 000 ks.

Tab. 6 Vstupní data příkladu s opt. velikostí obj. při nespojitém objednacím množství

Údaje	Označení	Hodnoty	Jednotky
Poptávka za období T	S	20 000	ks
Náklady na vyřízení objednávky	n_j	10 500	Kč
Procentuální náklady na skladování	n_s	17	% z průměrné zásoby za rok
Cena za jednotku	c	3 200	Kč/ks
Období	T	1	rok

Výpočet množství je následující:

$$Q = \frac{2Sn_j}{Tcn_s} = \frac{2 \cdot 20\,000 \cdot 10\,500}{1 \cdot 3\,200 \cdot 0,17} = 772\,000$$

Pro správné rozhodnutí je třeba splnit kritérium:

$$Q(Q + q) \geq Q \geq Q(Q - q)$$

Postupným výpočtem je zjištěno, jaký interval vyhovuje kritériu. Výsledek je uveden v následující tabulce.

Tab. 7 Výpočetní tabulka pro nespojité objednacím množství

Q	250	500	750	1 000
$Q(Q - q)$	0	125 000	375 000	750 000
$Q(Q + q)$	125 000	375 000	750 000	1 250 000

Z tabulky je patrné, že hodnota kritéria je splněna při objednacím množství $Q = 1\,000$ kusů.

Je nutné dopočítat náklady:

$$N(1\,000) = \frac{Q}{2} T n_s c + \frac{S}{Q} n_j = \frac{1\,000}{2} \cdot 1 \cdot 0,17 \cdot 3\,200 + \frac{20\,000}{1\,000} \cdot 10\,500 = 482\,000 \text{ Kč}$$

Tab. 8 Výsledek příkladu s opt. velikostí obj. při nespojitém objednacím množství

Údaje	Označení	Hodnoty	Jednotky
Optimální objednacím množství	Q_{opt}	1 000	ks
Náklady při určeném množství	$N(Q_{opt})$	482 000	Kč
Perioda objednání	t_c	18	dny

5.5 Sleva typu A

Poptávka po jistém produktu byla stanovena na 25 000 ks za rok. Skladovací náklady jsou 16% z průměrné zásoby v Kč za rok, náklady na jedno pořízení 5 500 Kč/obj. Ceník dodavatele zachycuje následující tabulka:

Tab. 9 Ceník k příkladu se slevou A

Interval	Spodní hranice Q [ks]	Hodní hranice Q [ks]	Cena za ks [Kč]
1	0	100	3 300
2	100	300	3 000
3	300	500	2 800
4	500	700	2 700
5	700	900	2 500
6	900	5 000	2 400

Tab. 10 Vstupní data k příkladu se slevou A

Údaje	Označení	Hodnoty	Jednotky
Poptávka za období T	S	25 000	Ks
Náklady na vyřízení objednávky	n_j	5 500	Kč
Procentuální náklady na skladování	n_s	16	% z průměrné zásoby za rok
Cena za jednotku	c	viz tabulka	-
Období	T	1	Rok

K vyřešení příkladu je třeba postupně vypočítat Q_{opt} při stanovených cenách v jednotlivých intervalech. Pokud se optimální množství nachází v tomto intervalu, je to zároveň i řešení, kolik kusů objednávat.

Ukázka výpočtu:

$$Q_{opt} = \sqrt{\frac{2Sn_j}{Tn_sc}} = \sqrt{\frac{2 \cdot 25\,000 \cdot 5\,500}{1 \cdot 0,16 \cdot 3\,300}} = 378 \text{ ks}$$

Tab. 11 Výpočetní tabulka k příkladu se slevou A

Interval	Spodní hranice Q [ks]	Hodní hranice Q [ks]	Cena za ks [Kč]	Q_{opt}
1	0	100	3 300	721
2	100	300	3 000	756
3	300	500	2 800	783
4	500	700	2 700	797
5	700	900	2 500	829
6	900	5 000	2 400	846

Zvýrazněný řádek jako jediný leží v intervalu, ke kterému náleží příslušná pořizovací cena.

Spočítají se náklady pro Q_{opt} v daném intervalu a ještě pro dolní hranici následujícího intervalu:

$$N(761) = \frac{Q}{2} \cdot T \cdot n_s \cdot c + \frac{S}{Q} \cdot n_j =$$

$$= \frac{761}{2} \cdot 1 \cdot 0,16 \cdot 2\,500 + \frac{25\,000}{761} \cdot 5\,500 \doteq 345\,000 \text{ Kč}$$

$$N(900) = \frac{1}{2} \cdot T \cdot n_s \cdot (n_j + Q \cdot c) + \frac{S}{Q} \cdot (n_j + Q \cdot c) =$$

$$= \frac{900}{2} \cdot 1 \cdot 0,16 \cdot 2\,400 + \frac{25\,000}{900} \cdot 5\,500 \doteq 325\,000 \text{ Kč}$$

Protože cena je výhodnější při vstupu do horního intervalu, který je nad intervalem nalezeným, jako optimální množství se zvolí dolní hranice vyššího intervalu s nižší cenou za kus. Tedy $Q_{opt} = 900$ ks.

Standardním způsobem zbývá vypočítat náklady a periodu pořízení:

$$t_c = \frac{Q_{opt} T}{S} = \frac{900}{25\,000} = 0,033 \text{ roku} \doteq 12 \text{ dní}$$

Tab. 12 Výsledky k příkladu se slevou A

Údaje	Označení	Hodnoty	Jednotky
Optimální objednávkové množství	Q_{opt}	900	ks
Náklady při určeném množství	$N(Q_{opt})$	325 000	Kč
Perioda objednání	t_c	12	dny

5.6 Sleva typu B

Poptávka po dětské panence byla stanovena na 1 000 ks za 1 rok. Skladovací náklady jsou 13% z průměrné zásoby v Kč za rok, náklady na jedno pořízení 250 Kč/obj. Ceník dodavatele je zachycen následující tabulkou:

Tab. 13 Ceník k příkladu se slevou B

Interval	Spodní hranice Q [ks]	Hodní hranice Q [ks]	Cena za ks [Kč]
1	0	150	210
2	150	300	205
3	300	750	200
4	750	5 000	198

V daných intervalech se počítají náklady spojené s objednáním celého množství z intervalu. Pokud optimální množství náležící k těmto nákladům spadá do příslušného intervalu, přejde se k výpočtu v intervalu dalším, kde se jednak spočítají náklady pro rozdíl horní a spodní hranice intervalu a dále se připočtou předchozí náklady. Postupuje se takto až do chvíle, kdy je optimální množství mimo tento interval. Předchozí interval a jeho optimální množství je pak hledaným výsledkem.

Tab. 14 Vstupní data k příkladu se slevou B

Údaje	Označení	Hodnoty	Jednotky
Poptávka za období T	S	1 000	Ks
Náklady na vyřízení objednávky	n_j	250	Kč
Procentuální náklady na skladování	n_s	13	% z průměrné zásoby za rok
Cena za jednotku	c	viz tabulka	-
Období	T	1	Rok

Ukázka výpočtu:

$$F_p = (Q - Q_{p-1})c_p + \sum_{s=1}^{p-1} (Q_s - Q_{s-1})c_s$$

$$Q_{p,opt} = \sqrt{\frac{2S(n_j + F_{p-1} - Q_{p-1}c_p)}{Tc_p n_s}}$$

Tab. 15 Výpočetní tabulka k příkladu se slevou B

Interval	Spodní hranice Q [ks]	Hodní hranice Q [ks]	Cena za ks [Kč]	$N(Q_{max} - Q_{min})$	F_i	Q_{opt}
1	0	150	210	31 500	31 500	136
2	150	300	205	30 750	62 250	274
3	300	750	200	90 000	152 250	439

Označený řádek je poslední, ve kterém ještě optimální množství spadá do příslušného intervalu.

Zbývá dopočítat náklady na objednávku a periodu objednání:

$$A_i = (n_j + F_{p-1} - Q_{p-1}c_p)c_i = 2\,500$$

$$N_p(Q_{p,opt}) = SC_p + 0,5Tn_s(F_{p-1} - Q_{p-1}c_p) + \sqrt{2STn_sA_p} = 79\,417 \text{ Kč}$$

$$t_c = \frac{Q_{opt}T}{S} = \frac{139}{(750/(27/12))} = 0,417 \text{ roku} = 5 \text{ měsíců}$$

Tab. 16 Výsledky k příkladu se slevou B

Údaje	Označení	Hodnoty	Jednotky
Optimální objednávací množství	Q_{opt}	139	ks
Náklady při určeném množství	$N(Q_{opt})$	79 417	Kč
Perioda objednání	t_c	5	měsíce

5.7 Q-systém

Firma Mák s.r.o. potřebuje řešit situaci s dopravou zásob. Musí se zabezpečit 36 000 ks balení máku na rok. Cena balení je 120 Kč. Skladovací náklady jsou $n_s = 10\%$ z průměrné zásoby za rok, pořizovací náklady jsou $n_j = 1\,200$ Kč/objednávka. Pořizovací lhůta je $d = 14$ dní. Předpokládaná směrodatná odchylka v rámci d je $\sigma_{Qd} = 200$ ks. Je třeba určit optimální velikost dodávky, související náklady, bod znovuobjednávky a velikost pojistné zásoby při požadované úrovni obsluhy 95%, resp. 99%. Jak se zvýší střední hodnota nákladů při zvýšení pojistné ztráty?

Tab. 17 Vstupní data k příkladu Q-systém

Údaje	Označení	Hodnoty	Jednotky
Poptávka za období T	S	36 000	Ks
Náklady na vyřízení objednávky	n_j	1 200	Kč
Procentuální náklady na skladování	n_s	20	% z průměrné zásoby za rok
Cena za jednotku	c	120	Kč
Období	T	1	Roky
Pořizovací lhůta	t_{vo}	15	Dny
Směrodatná odchylka v rámci d	σ_{Qd}	200	Ks

Nejprve se tedy vypočítá optimální velikost objednávky a k ní související náklady:

$$Q_{opt} = \sqrt{\frac{2Sn_j}{Tn_sc}} = \sqrt{\frac{2 \cdot 36\,000 \cdot 1\,200}{1 \cdot 0,20 \cdot 120}} = 1\,900 \text{ ks}$$

$$N(Q_{opt}) = \sqrt{2STcn_sn_j} = \sqrt{2 \cdot 36\,000 \cdot 1 \cdot 120 \cdot 0,2 \cdot 1\,200} = 45\,600 \text{ Kč}$$

Optimální velikost jedné objednávky je tedy 1 900 ks a náklady na tuto optimální objednávku jsou 45 600 Kč. Očekávanou poptávku je nutné vypočítat podle stanovené pořizovací lhůty (15 dní = $\frac{15}{365}$ roku). Očekávaná poptávka za toto období r^* je 1 480 ks ($36\,000 \cdot \frac{15}{365}$). Pro výpočet se použijí tabulky normovaného normálního rozdělení, ze kterých pro úroveň obsluhy $\gamma = 0,95$ vychází hodnota 1,645. Pro $\gamma = 0,99$ vychází hodnota 2,327.

Pro udržení úrovně obsluhy na 95% je třeba vytvořit pojistnou zásobu w ve výši:

$$w \geq z^* \sigma_{Qd} = 1,645 \cdot 200 = 329 \text{ ks}$$

Pro udržení úrovně obsluhy na 99% je třeba vytvořit pojistnou zásobu w ve výši:

$$w \geq z^* \sigma_{Qd} = 2,327 \cdot 200 = 466 \text{ ks}$$

V dalším kroku se vypočítá bod znovuoobjednávky:

Pro udržení úrovně obsluhy na 95%:

$$\text{Signální stav zásob je: } r^* + w = 1\,480 + 329 = 1\,809 \text{ ks}$$

$$\text{Náklady na skladování se zvýší o } 329 \cdot (120 \cdot 0,2) = 7\,896 \text{ Kč}$$

Pro udržení úrovně obsluhy na 99%:

$$\text{Signální stav zásob je: } r^* + w = 1\,480 + 466 = 1\,946 \text{ ks}$$

$$\text{Náklady na skladování se zvýší o } 466 \cdot (120 \cdot 0,2) = 11\,184 \text{ Kč}$$

Tab. 18 Výsledku k příkladu Q-systém

Údaje	Označení	Hodnoty pro $\gamma = 0,95$	Hodnoty pro $\gamma = 0,99$	Jednotky
Velikost objednávek	Q_{opt}	1 900	1 900	ks
Náklady při určen. množství	$N(Q_{opt})$	53 496	56 784	Kč
Pojistná zásoba	w	329	466	ks
Signální stav zásob	$r^* + w$	1 809	1 946	ks

5.8 P-systém

Firma Mléko s.r.o. potřebuje vyřešit situaci. Je nutné zabezpečit $S = 36\,000$ ks kartonů mléka na rok. Cena za karton je 120 Kč. Skladovací náklady jsou $n_s = 10\%$ z průměrné zásoby za rok, pořizovací náklady jsou $n_j = 1\,200$ Kč/objednávka. Pořizovací lhůta je $d = 14$ dní. Předpokládaná směrodatná odchylka v rámci d je $\sigma_{Qd} = 200$ ks. Je třeba určit optimální velikost dodávky, související náklady, bod znovuobjednávky a velikost pojistné zásoby při požadované úrovni obsluhy 95%, resp. 99%. Jak se zvýší střední hodnota nákladů při zvýšení pojistné ztráty?

Tab. 19 Vstupní data k příkladu P-systém

Údaje	Označení	Hodnoty	Jednotky
Poptávka za období T	S	36 000	ks
Náklady na vyřízení objednávky	n_j	1 200	Kč
Procentuální náklady na skladování	n_s	20	% z průměrné zásoby za rok
Cena za jednotku	c	120	Kč
Období	T	1	roky
Pořizovací lhůta	t_{vo}	15	dny
Směrodatná odchylka v rámci d	σ_{Qd}	200	ks

Nejprve se tedy vypočítá optimální velikost objednávky a čas objednání:

$$Q_{opt} = \sqrt{\frac{2Sn_j}{Tn_sc}} = \sqrt{\frac{2 \cdot 36\,000 \cdot 1\,200}{1 \cdot 0,20 \cdot 120}} = 1\,900 \text{ ks}$$

$$t_c = \frac{T}{S/Q_{opt}} = \frac{1}{36\,000/1\,900} = 0,0528 \text{ roku} = 19,3 \text{ dní}$$

$$\sigma_{t_{vo}+t_c} = \sqrt{(t_{vo} + t_c)\sigma_{Qd}^2} = \sqrt{\left(\frac{15}{365} + 0,0528\right) 200^2} = 61,285$$

$$z_{95} = 1,64485362695147 \quad z_{99} = 2,32634787404084$$

$$\begin{aligned} x_{h95} &= S(t_{vo} + t_c) + z_{95}\sigma_{t_0+t_c} = 36\,000 \left(\frac{15}{365} + 0,0528 \right) + 1,645 \cdot 61,285 = \\ &= 3\,380,25 + 100,81 = 3\,481 \text{ ks} \end{aligned}$$

$$\begin{aligned} x_{h99} &= S(t_{vo} + t_c) + z_{99}\sigma_{t_0+t_c} = 36\,000 \left(\frac{15}{365} + 0,0528 \right) + 2,326 \cdot 61,285 = \\ &= 3\,380,25 + 142,55 = 3\,523 \text{ ks} \end{aligned}$$

Tab. 20 Výsledky k příkladu P-systém

Údaje	Označení	Hodnoty pro $\gamma = 0,95$	Hodnoty pro $\gamma = 0,99$	Jednotky
Perioda objednávek	t_c	19,3	19,3	dny
Pojistná zásoba	w	101	143	ks
Horní objednáací hranice	x_h	3 481	3 523	ks

6 POUŽITÉ ŘEŠENÍ

Systém řízení zásob je implementován v programovacím jazyce Python. Díky obsáhlé podpoře vývojářů třetích stran bylo možné vytvořit program s grafickým prostředím, který je dostačující pro implementaci dané tematiky.

6.1 Python

Python je programovací jazyk podporující více paradigmat. Je tedy multiparadigmatický. V roce 1991 jej navrhl Guido van Rossum. Jeho značná výhoda je, že je vyvíjen jako open source (počítačový software s otevřeným zdrojovým kódem). Je dynamicky interpretovaný, často využívaný jako skriptovací jazyk. V Pythonu se ovšem dají navrhovat rozsáhlé aplikace, nejenom skrze objektově orientovaný přístup, využívající grafického rozhraní. Hlavní výhodou tohoto jazyka je jednoduchost, se kterou je možné se ho naučit a přehlednost kódu, kterou sám program podporuje, protože je nutné podřízené bloky a podmínky obsazovat. Program na rozdíl od většiny programovacích jazyků nepoužívá oddělovače, pouze odsazení a posun kódu na nový řádek.

Hlavním důvodem, proč byl Python vybrán pro tuto práci, jsou mé osobní zkušenosti s ním a také to, že podporuje knihovny třetích stran, které umožňují danou tematiku efektivně zpracovat. Jedním z hledisek je, že pomocí jedné z knihoven je možné celou aplikaci převést do spustitelného tvaru na systémech, které nemají podporu Pythonu a dalších přidružených knihoven, tedy do souborové struktury obsahující soubor typu *executable* na operačních systémech Windows. Díky tomu je aplikace využitelná bez jakékoliv instalace, pouze stačí spustit soubor s koncovkou *exe*. Ani aplikace sama nevyžaduje instalaci.

K vývoji aplikace je využit Python 3.4.

6.2 Knihovny třetích stran

K vytvoření aplikace bylo třeba využít hned několik knihoven, které rozvinuly různé funkce jazyka Python. Ať už grafické prostředí, výpočetní funkce, vykreslování grafů a matematických formulí nebo převod do spustitelné aplikace.

Tkinter

Jeho název vychází ze spojení částí Tk a interface (česky rozhraní k Tk). Jedná se o modul, který vytváří vrstvu nad grafickou knihovnou Tk/Tcl. Jeho funkcí je vytvářet grafická okna tak, jak jsou známé z operačních systémů Windows, Linux a Mac. Programy vytvořené pomocí tohoto grafického modulu fungují na všech těchto platformách.

matplotlib

Tato vykreslovací knihovna pro Python je hojně využívána k grafickému zobrazení mnoho druhů grafů, a to jak rovinných, tak prostorových. Díky této knihovně se dá také využít vypsování vzorců ve vzhledu podobnému sazečskému programu LaTeX.

NumPy a SciPy

Jedná se o dvě knihovny, které se starají o matematické výpočty. Knihovnu NumPy vyžaduje i knihovna matplotlib. Knihovna SciPy je využita například pro generování náhodných hodnot normálního rozdělení.

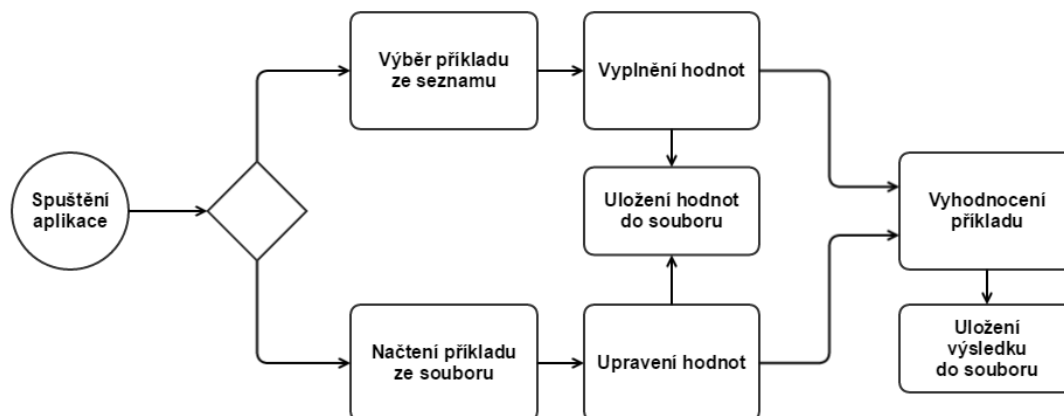
Další knihovny

Některé další knihovny jsou vyžadovány knihovnami dříve uvedenými, a proto je nutné je při vytváření programu doinstalovat. Jedná se například o knihovny `six` a `python-dateutil`.

7 PROGRAM

7.1 Popis programu

Program je navržen podle následujícího schématu, ze kterého je patrná jeho funkce:



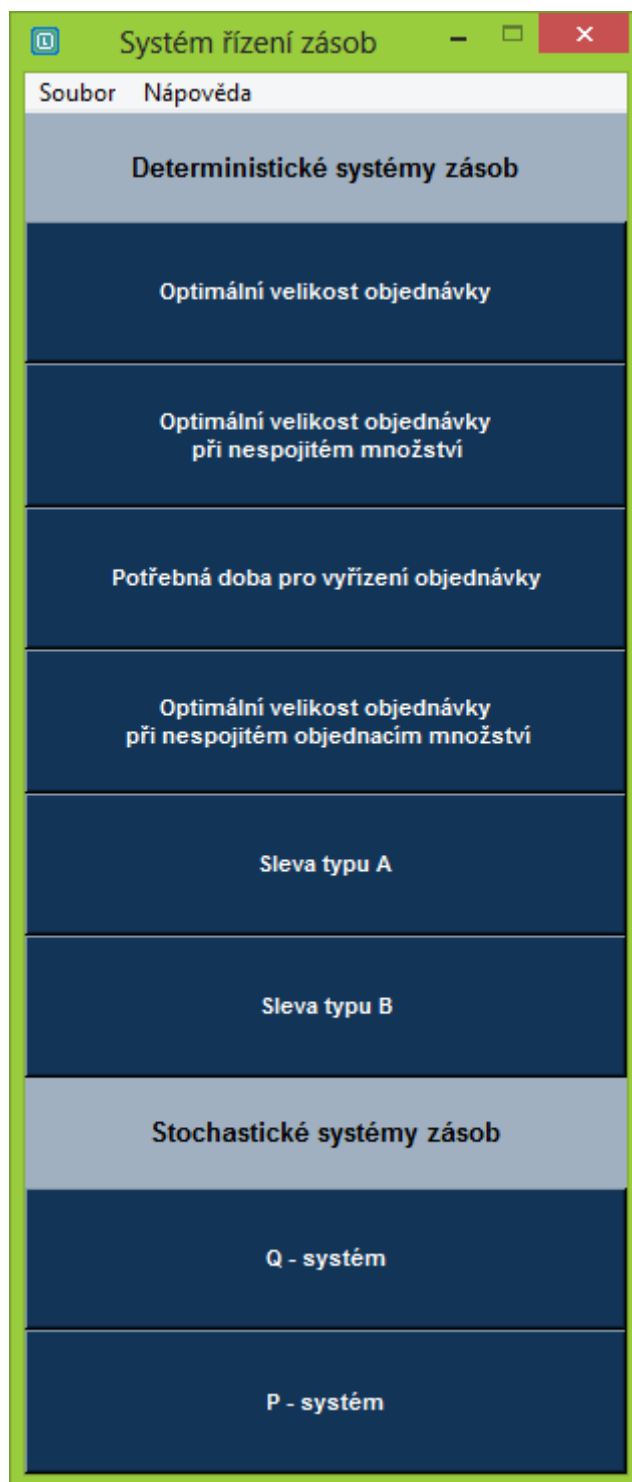
Obr. 1 Schéma programu

Po spuštění aplikace se dají v úvodním okně načíst uložené hodnoty ze souboru, které je poté možné editovat v okně konkrétního příkladu, nebo zvolit příklad nový a zadávat hodnoty od začátku. V každém kroku vyplňování mohou být hodnoty uloženy, i když jsou neúplné nebo editované. Pokud jsou hodnoty korektně vyplněny, je možné potvrdit vyhodnocení.

Po stisknutí tlačítka „Vyhodnotit“ je při korektně zadaných datech zobrazen graf a matematický zápis příslušný ke konkrétnímu příkladu.

Toto okno může být uloženo do souboru (formát pdf, png, ...). Také je možné volně manipulovat s grafem, tedy jej libovolně posouvat a přibližovat.

Po spuštění se program dostane do hlavního menu, které je následující:

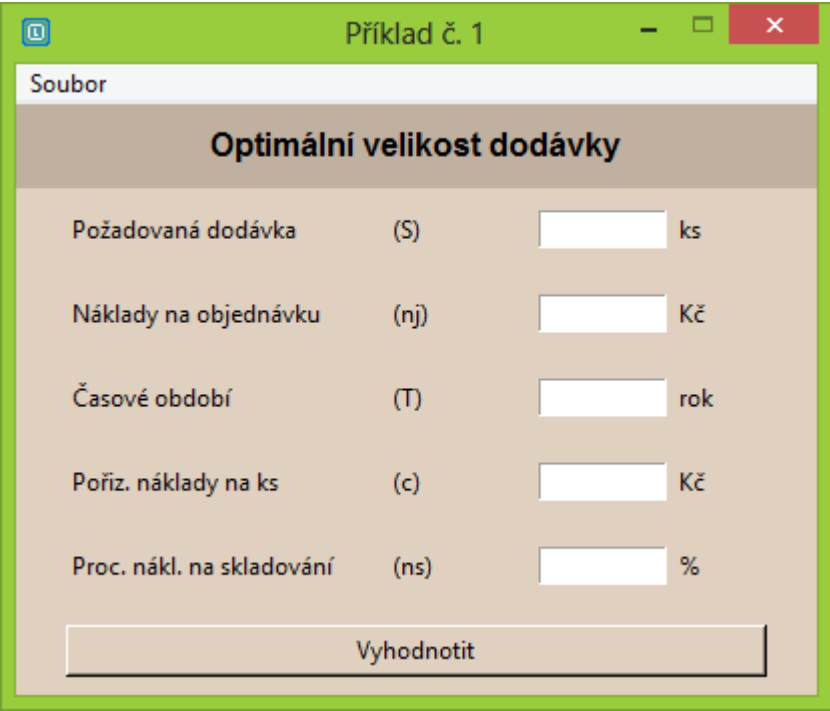


Obr. 2 Menu hlavního programu

Popisy úloh slouží jako tlačítka a při stisknutí se otevře nové okno s danou úlohou.

V menu „Soubor“ je možné aplikaci zavřít, nebo z něj otevřít již existující úlohu uloženou v souboru použitelném pro aplikaci.

Po stisknutí tlačítka „Optimální velikost objednávky“ se otevře následující podokno aplikace:



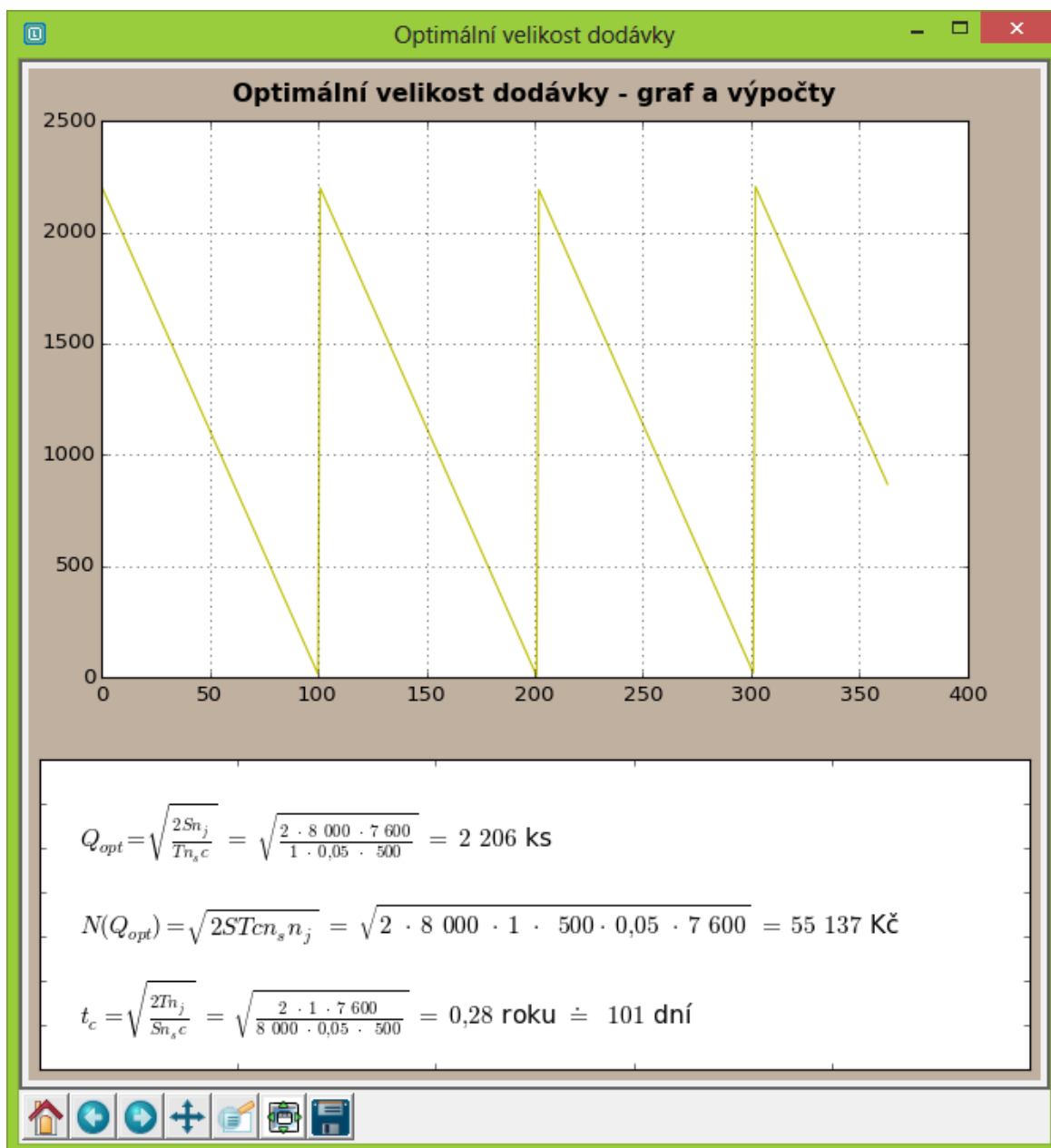
The screenshot shows a window titled "Příklad č. 1" with a green header bar. Below the header is a menu bar with "Soubor". The main content area has a title "Optimální velikost dodávky" and contains five input fields with labels and units:

Label	Symbol	Unit
Požadovaná dodávka	(S)	ks
Náklady na objednávku	(nj)	Kč
Časové období	(T)	rok
Pořiz. náklady na ks	(c)	Kč
Proc. nákl. na skladování	(ns)	%

At the bottom of the form is a button labeled "Vyhodnotit".

Obr. 3 Ukázkový vstup programu

V tomto okně se zadávají známé hodnoty k příkladu, které jsou po stisku tlačítka „Vyhodnotit“ zkontrolována (zda se jedná o čísla), a pokud je vše bez chyby, použijí se pro výpočet. Po výpočtu je zobrazeno další okno s grafem a použitými výpočty:



Obr. 4 Ukázkový výstup programu

Funkcionalita je tedy zřejmá. Další příklady v programu fungují podobně, pouze se zadávají odlišné hodnoty a výsledky jsou přirozeně také odlišné.

7.2 Zdrojový kód

Zdrojový kód zde uvedený pochází z úvodu programu a z prvního příkladu aplikace. Ostatní příklady jsou obdobou příkladu prvního.

Hlavní okno

Hlavní okno je vytvořeno objektově. Prvotní rozložení je realizováno následujícím kódem:

```
def __init__(self, master=None):
    Frame.__init__(self, master)
    self.grid()
    self.master.title("Systém řízení zásob")
    self.master.resizable(width=FALSE, height=FALSE)
    self.master.geometry('{ }x{ }+{ }+{ }'.format(300, 700, 72, 10))
    self.master.iconbitmap('favicon.ico')
```

Menu

Pro sestavení menu je použit následující kód:

```
hlavniMenu = Menu(root)

# vytvořit rozbalovací menu a přidat ho k hlavnímu menu
menuSoubor = Menu(hlavniMenu, tearoff=0)
menuSoubor.add_command(label="Otevřít", command=openDialog)
menuSoubor.add_separator()
menuSoubor.add_command(label="Ukončit program", command=root.quit)
hlavniMenu.add_cascade(label="Soubor", menu=menuSoubor)

# další menu
menuNapoveda = Menu(hlavniMenu, tearoff=0)
menuNapoveda.add_command(label="O aplikaci", command=aboutApp)
hlavniMenu.add_cascade(label="Nápověda", menu=menuNapoveda)

# zobrazení menu
root.config(menu=hlavniMenu)
```

Vytvoření okna příkladu

Následující funkci využívá každý příklad k tomu, aby vytvořil okno, do kterého bude dodávat údaje a formuláře:

```
def vytvorOkno(self, title, width, height):
    # nastavení objektu hlavního okna
    window = Toplevel(self)
    # nastavení titulu hlavního okna
    window.wm_title(title)
    # nastavení rozměrů hlavního okna
    window.geometry('{ }x{ }'.format(width, height))
    # nastavení změny nastavení (změna zakázána)
    window.resizable(width=FALSE, height=FALSE)
    # nastavení zaměření okna
    window.focus_force()
```

```

        # nastavení ikony
        window.iconbitmap('favicon.ico')
        # návratová hodnota obsahující vytvořený objekt s oknem (pro
vyplnění okna)
        return(window)

```

Funkce přebírá parametry *self* (pro manipulaci v rámci třídy), *title* (pro nastavení titulu okna), *width* a *height* (oba pro nastavení velikosti okna – šířky a délky). Navíc je pro potřeby aplikace zakázána změna rozměrů okna, které je vytvořené přesně v daném rozměru. Při vytvoření se navíc okno zaměří. Jako předposlední příkaz je zde uvedeno přiřazení souboru s ikonou, která se zobrazí v titulu okna. Posledním příkazem je pak vytvořený objekt navracen ke zdroji volání funkce, kde se naváže na proměnnou a je s ním dále pracováno (například se vyplní dalšími údaji a formuláři).

Načtení hodnot ze souboru

Pro načtení ze souboru je vytvořena funkce s názvem *openDialog*, která je vyvolána poté, co uživatel zvolí z menu „Soubor“ volbu „Otevřít“. Funkce nepřijímá parametry, protože je volána po stisku tlačítka z menu. Funkce tedy vypadá následovně:

```

# funkce pro načtení uložených příkladů
def openDialog():
    # inicializace proměnné příklad kvůli testování
    priklad = 0

    # otevření dialogu a přiřazení cesty k souboru do proměnné
    loadFile = askopenfilename(parent=root, filetypes=[('Lsystem', '.lsys')])

    # vytvoření slovníku, do které se budou ukládat proměnné
    values = {}

```

Proměnná *priklad* je použita v dalším kódu k testování a je třeba ji nastavit na výchozí hodnotu, aby byl test platný i při negativním výskytu názvu příkladu v těle souboru. Do proměnné *loadFile* se uloží cesta vybraná přes dialogové okno určené pro výběr souboru. Program načítá soubory *LSystem*, které mají koncovku *.lsys*. *Values* je zde nastaveno jako slovník, do kterého se budou postupně při čtení souboru plnit identifikované hodnoty ze souboru. Posléze budou předány ke zpracování. Kód pokračuje následovně, zkrácen o jedno odsazení:

```

# pokud byl vybrán soubor, provede se otestování
if loadFile:

    # v souboru se projde každý řádek
    for line in open(loadFile):
        # první řádek obsahuje název příkladu, podle kterého je identifikován
        if line.replace('\n', '') == '---Optimální velikost objednávky---':
            priklad = '01'

            ... (zkrácení kódu) ...

        elif line.replace('\n', '') == '---Q - systém---':
            priklad = '07'
        elif line.replace('\n', '') == '---P - systém---':
            priklad = '08'

```

Pokud je nastavena hodnota v proměnné *loadFile* rozdílná od 'None', tedy je vybrán soubor, provede se projití tohoto souboru po řádcích. V prvním řádku uloženého souboru je uveden název příkladu a podle toho jsou pak načítány další parametry na dalších řádcích souboru. Následující test vyhodnotí, jestli je nastaven příznak příkladu:

```
else:
    if priklad == '01':
```

Je-li vyhodnocení podmínky kladné, začnou se identifikovat řádky s příslušnými proměnnými a zapisovat do slovníku *values*. Bez příslušného odsazení je to realizováno následujícím kódem:

```
if line[:2] == 'ns':
    values['ns'] = line.split('\t')[1].replace('\n', '')
if line[:1] == 'T':
    values['T'] = line.split('\t')[1].replace('\n', '')
if line[:1] == 'S':
    values['S'] = line.split('\t')[1].replace('\n', '')
if line[:2] == 'nj':
    values['nj'] = line.split('\t')[1].replace('\n', '')
if line[:1] == 'c':
    values['c'] = line.split('\t')[1].replace('\n', '')
if line[:4] == 'graf':
    values['graf'] = line.split('\t')[1].replace('\n', '')
```

Hodnoty jsou v souboru uloženy v řádcích, kdy první hodnota na řádku je proměnná, která je identifikovaná pomocí porovnání s očekávanými hodnotami. Pokud je hodnota shodná se vzorem, dojde k rozdělení řádku pomocí funkce *split* a druhá část se uloží do slovníku se správným označením.

Jako poslední zbývá načtené hodnoty předat funkci, která se postará o vytvoření okna a zpracuje předané parametry:

```
if priklad == '01':
    self.example_01(values)
elif priklad == '02':
    self.example_02(values)

    ... (zkrácení kódu) ...

elif priklad == '07':
    self.example_07(values)
elif priklad == '08':
    self.example_08(values)
```

Funkce, která zajistí zpracování hodnot a vytvoří okna se zadáváním je zpracována k ukázkovému příkladu v dalším textu.

Ulož zadání

K uložení zadání se používá jednoduchá funkce:

```
def ulozZadani(self, title, variables, window):
    # přes dialogové okno se vybere místo pro uložení a název souboru
    fileToSave = asksaveasfilename(parent=window, filetypes=
    [('Lsystem', '.lsys')], defaultextension='lsys')

    # pokud se dialog zavře, nic se neprovede
    if fileToSave:
        # otevře se soubor pro zápis
        outputFile = open(fileToSave, "w")
        # zapíše se hlavička příkladu
        outputFile.write('---' + title + '---\n')
        # postupně se zapíší všechny uvedené hodnoty ve slovníku
        for piece in variables:
            outputFile.write(piece+'\t'+variables[piece]+'\n')

        # soubor se zavře
        outputFile.close()
```

K výběru souboru je použito dialogové okno, které dovoluje ukládat soubory s koncovkou `.lsys`. Tato koncovka se používá pro ukládání příkladů. V případě, že není vybrán žádný soubor, funkce končí. V opačném případě je soubor vybrán, otevře se v režimu zápisu pomocí funkce `open` s volitelným atributem v režimu `w` (writable). Jako první se zapíše hlavička, která obsahuje název souboru a slouží pro pozdější identifikaci příkladu. Poté se spustí cyklus, který jednotlivé hodnoty ze slovníku, který byl funkci předán, zapíše na řádky tak, aby první hodnota odpovídala identifikátoru a druhá obsahu. Po zapsání všech proměnných je soubor uzavřen a uložen.

Číslo do zápisu

Protože knihovna `matplotlib` vypisuje čísla v textovém formátu (nemají oddělovač tisíců), nabízí se tento výstup kvůli přehlednosti upravit. Vytvořená funkce k tomuto účelu má následující podobu:

```
# funkce pro zapis do vysledku
def cisloDoZapisu(self, cislo, desetiny = 0):
    #promenne
    result = ""
    des = ""
    #rozdeleni na cele a desetinne
    if "." in str(cislo):
        cel, des = str(cislo).split(".")
    elif "," in str(cislo):
        cel, des = str(cislo).split(",")
    else:
        cel = str(cislo)

    # úprava při nulách na začátku
    while cel[0] == '0' and len(cel) > 1:
        cel = cel[1:]

    # výpočet pro zápis čísla po třech číslicích
    i = 3 - len(cel) % 3

    # postupný výpis
    for letter in cel:
        if (i)%3 == 0 and i != 0:
            result += "\/"
        result += letter
```

```

        i += 1

# úprava desetiny
if des and desetiny > 0:
    if len(des) < desetiny:
        desetiny = len(des)
    result += "," + des[:desetiny]
# vrácení výsledku
return(result)

```

Funkce rozdělí číslo na celou část a desetinnou část. Na celé části provede takové úpravy, které do výstupu vloží oddělovač tisíců. Desetinná část oddělovač nemá, je připojena pouze v případě, že je uvedena ve volitelném parametru. Hodnota parametru určuje, kolik desetinných míst se za celou částí vypíše.

Ukázkový příklad

Kód příkladu je rozdělen na několik částí, které plní svůj specifický účel. Pro potřeby zobrazení je vytvořena vnitřní funkce objektu hlavního okna. Pro každý příklad je vytvořeno specifické okno, které obsahuje relevantní informace s ohledem na daný příklad (tento objekt může být vytvořen buďto z hlavního menu, nebo po načtení ze souboru). Dále je pak vytvořen záznam v sekci pro ukládání již zadáných hodnot a jako poslední část je v programu vytvořena oblast, která zadané hodnoty realizuje a vytvoří výstupní okno.

Vstupní okno k příklad *P-systém* vypadá následovně:

P - systém			
Požadovaná dodávka	(S)	36000	ks
Náklady na objednávku	(nj)	1200	Kč
Časové období	(T)	1	rok
Pořiz. náklady na ks	(c)	120	Kč
Proc. nákl. na skladování	(ns)	20	%
Doba dodání objednávky	(tvo)	15	dny
Směrodatná odchylka	(oQd)	200	ks
Úroveň obsluhy	(y)	0.95	
Vyhodnotit			
Interval grafu v násobcích T		1	

Obr. 5 Vyplněný vstup příkladu *P-systém*

Kód k vytvoření tohoto okna je následující:

```
def example_08(self, values = None):
    # určení názvu příkladu
    nazev = r'P - systém'
    # nastavení parametrů okna (název, šířku, délku)
    e01Window = self.vytvorOkno(nazev, 440, 445)
```

Tento kus kódu slouží k vytvoření vnitřní funkce. V seznamu parametrů je použito *self*, což je standardní využití hodnot z objektu. Jako další je zde uveden parametr *values = None*, a to z toho důvodu, zůstane-li nevyplněn, nepoužije se. Pokud je vyplněn, předává oknu hodnoty načtené ze souboru. Dále je zde uveden název příkladu, který se používá pro pozdější identifikaci. Každý příklad má samozřejmě svůj specifický název. Dále se zavolá funkce *self.vytvorOkno* pro vytvoření okna (viz výše) s nastavením názvu a rozměrů.

Nyní je tedy vytvořen objekt okna sloužícího k nastavení hodnot příkladu. Dále je třeba doplnit informace spolu s formuláři pro zadávání hodnot. K tomu je vytvořena vnořená funkce s názvem *naplnOkna*:

```
# funkce, která je použita pro náplň vytvořeného okna (okno, název)

def naplnOkna(window, nazev):

    # počet řádků okna
    for row in range(11):
        # nastavení minimální velikosti řádku (použito, aby byl každý
        # řádek stejně tlustý)
        window.rowconfigure(row, weight=1, minsize=35)

    # počet sloupců okna
    for column in range(8):
        # nastavení každého jednotlivého sloupce
        window.columnconfigure(column, weight=1)
```

Funkce má nastaven počet řádků a sloupců, které postupně projde a nastaví jim formátovací parametry. Pro sloupce to je pouze nastavení stejné váhy, kdy se při nezadaných údajích ve sloupci roztáhnou na stejnou šířku. Pro řádky je to podobné, analogicky se při nezadaných údajích roztáhnou na stejnou šířku. Navíc zachovávají minimální šířku a to 35 pixelů.

Rozdělení na řádky a sloupce má své opodstatnění, protože zadání příkladu je rozděleno do mřížky. V takto nastavené mřížce se na každou buňku dá odkazovat uvedením příslušného řádku a sloupce. Takto budou nastavovány parametry pro většinu buněk.

Další kód je až do odvolání obsahem funkce *naplnOkna*, pro větší přehlednost a uvolnění místa pro zápis je pokračování funkce uváděno bez odsazení, i když v kódu je samozřejmě odsazení nezbytné.

Pro grafické přizpůsobení je navíc využito rámce, který je možné nastavit jednotně pro více buněk a nastavit mu příslušné parametry:

```
# vytvoření objektu Frame, který vytváří seskupení v okně (okno, barva
# pozadí)
FrameWindow = Frame(window, bg="#EEEEEE")

# nastavení prvního řádku a sloupce, dále jeho rozměr a přichycení k okrajům
```

```
FrameWindow.grid(row = 1, column = 0, rowspan = 6, columnspan = 8, sticky
= W+E+N+S)
```

Z kódu je patrné, že se inicializuje rámec, který se naváže na příslušné okno a rovnou se nastaví jeho barva. Po inicializaci se nastaví další vlastnosti, v tomto případě jeho zahájení (druhý řádek, první sloupec) a rozměr (6 řádku a 8 sloupců). Navíc je také nastaveno přichycování k okrajům. To znamená, že se v rámci buňky roztáhne přes celou její plochu. Protože je zde parametr vidět poprvé, nabízí se rozvést jeho nastavení. Je využita analogie ke světovým stranám:

W (west – západ) – přichytí se na levou hranu buňky
 E (east – východ) – přichytí se na pravou hranu buňky
 N (north – sever) – přichytí se na horní hranu buňky
 S (south – jih) – přichytí se na dolní hranu buňky

Jako poslední před tím, než se začne vyplňovat okno údaji, je vytvoření a nastavení objektů potřebných pro použití formulářů:

```
# vytvoření objektů Entry sloužících pro vyplnění hodnot
Form_S = Entry(window, width = 10, justify=RIGHT)
Form_nj = Entry(window, width = 10, justify=RIGHT)
Form_T = Entry(window, width = 10, justify=RIGHT)
Form_c = Entry(window, width = 10, justify=RIGHT)
Form_ns = Entry(window, width = 10, justify=RIGHT)
Form_tvo = Entry(window, width = 10, justify=RIGHT)
Form_osmer = Entry(window, width = 10, justify=RIGHT)
Form_obsl = Entry(window, width = 10, justify=RIGHT)
Form_graf = Entry(window, width = 10, justify=RIGHT)
```

Každá proměnná je vytvořena s názvem reflektujících hodnotu z příkladu, tedy následovně podle popisu:

Form_S – očekávaná poptávka
Form_nj – náklady na vyřízení objednávky
Form_T – časové období
Form_c – pořizovací cena
Form_ns – procentuální náklady na skladování
Form_tvo – doba potřebná pro vyřízení objednávky
Form_osmer – směrodatná odchylka poptávky
Form_obsl – požadovaná úroveň obsluhy
Form_graf – počet dní zobrazených na grafu v násobcích *T*

Objekt se vždy vytvoří s užitím v určitém okně (parametr *window*), s určenou šířkou (parametr *width*) a se zarovnáním vpravo (parametr *justify* = *RIGHT*).

V předchozím kódu, při vytvoření funkce třídy, byl zmíněn parametr *values*, který se v této části otestuje:

```
if values:
    if 'S' in values.keys():
        Form_S.insert(0, values['S'])
    if 'nj' in values.keys():
        Form_nj.insert(0, values['nj'])
    if 'T' in values.keys():
        Form_T.insert(0, values['T'])
    if 'c' in values.keys():
```

```

        Form_c.insert(0, values['c'])
    if 'ns' in values.keys():
        Form_ns.insert(0, values['ns'])
    if 'dtvo' in values.keys():
        Form_tvo.insert(0, values['dtvo'])
    if 'graf' in values.keys():
        Form_graf.insert(0, values['graf'])
    if 'osmer' in values.keys():
        Form_osmer.insert(0, values['osmer'])
    if 'obsl' in values.keys():
        Form_obsl.insert(0, values['obsl'])
    else:
        Form_graf.insert(0, '1')
else:
    Form_graf.insert(0, '1')

```

Pokud byl funkci parametr předán, jsou v něm uvedené hodnoty nahrané ze souboru pomocí funkce uvedené výše (*openDialog*).

Další následuje už samotné vyplňování dat do okna.

První je vložen úvodní titulek na první řádek a šířku celého okna:

```

# název okna vytvoření pomocí objektu Label
Label(window, bg='#BBBBBB', text=nazev, font="Arial 11 bold",
height="1",width=380).grid(row = 0, column = 0, columnspan = 6, sticky =
W+E+N+S)

```

Jako u předchozích prvků jsou u něj určené atributy, navíc se zde vyskytuje nastavení fontu a text je přiřazen z proměnné *nazev*.

Aby bylo možné atributy prvků měnit podle potřeby hromadně, jsou zde dva vytvořené slovníky, ve kterých stačí změnit požadovaný atribut, který je navázán na objekty typu *Label* nebo *Entry*:

```

# parametry prvků
labelProperties = {'bg':'#EEEEEE', 'sticky':'w'}
formProperties = {'sticky':''}

```

Ted' je již možné vložit většinu zbývajících prvků do vytvořeného objektu okna se zadáním příkladu:

```

# řádek s očekávanou poptávkou
Label(window, bg=labelProperties['bg'], text=u"Očekávaná poptávka").grid
(row=1, column = 1, sticky=labelProperties['sticky'])

Label(window, bg=labelProperties['bg'], text=u"(S)").grid(row=1, column =
2, sticky=labelProperties['sticky'])

Form_S.grid(row=1, column=3, sticky=formProperties['sticky'])

Label(window, bg=labelProperties['bg'], text=u" ks").grid(row=1, column =
4, sticky=labelProperties['sticky'])

# řádek s náklady
Label(window, bg=labelProperties['bg'], text=u"Náklady na objednávku")
.grid (row=2, column = 1, sticky=labelProperties['sticky'])

```

```

Label(window, bg=labelProperties['bg'], text=u"(nj)").grid(row=2, column =
2, sticky=labelProperties['sticky'])

Form_nj.grid(row=2, column=3, sticky=formProperties['sticky'])

Label(window, bg=labelProperties['bg'], text=u" Kč").grid(row=2, column =
4, sticky=labelProperties['sticky'])

... (zkrácení kódu) ...

# řádek s úrovní obsluhy
Label(window, bg=labelProperties['bg'], text=u"Úroveň
obsluhy").grid(row=8, column = 1, sticky=labelProperties['sticky'])

Label(window, bg=labelProperties['bg'], text=u"(y)").grid(row=8, column =
2, sticky=labelProperties['sticky'])

Form_obs1.grid(row=8, column=3, sticky=formProperties['sticky'])

Label(window, bg=labelProperties['bg'], text=u"").grid(row=8, column = 4,
sticky=labelProperties['sticky'])

```

Každý řádek je vytvořen za čtyř částí. Popis proměnné, zkratka proměnné, formulář na vyplnění hodnoty a jednotka proměnné.

Následuje potvrzovací tlačítko, které je nastavené tak, aby při stisku vyhodnotilo zadané hodnoty a předalo je další funkci:

```

# tlačítko vyhodnocení
Button(window, text=u"Vyhodnotit", width=200, bg="#CFCFCF",
command=doExample).grid(row=9, column = 1, columnspan = 4)

```

Ve vlastnostech tlačítka se poprvé objevuje atribut *command*. Ten slouží k tomu, aby mu byla předávána funkce, kterou posléze spustí. Jméno funkce je zadáváno bez složených závorek a je bez atributů. Kdyby byly závorky zadány, funkce by se provedla ihned při inicializaci prvku tlačítka. Funkce *doExample* je popsána dále.

Jako poslední ze zobrazovaných informací je vložen řádek s formulářem, ve kterém je možné nastavit, kolik dní se na grafu zobrazí v násobcích *T*:

```

# násobky T
Label(window, bg=labelProperties['bg'], text=u"Interval grafu v násobcích
T").grid(row=10, column = 1, columnspan=2,
sticky=labelProperties['sticky'])
Form_graf.grid(row=10, column=3, sticky=formProperties['sticky'])

```

Pokud není hodnota zadána jinak, načte se při vytváření nového příkladu, nebo při načtení ze souboru na hodnotu 1.

K vyhodnocení zadaných hodnot slouží funkce *doExample*, která je spuštěna po stisku tlačítka vyhodnocení:

```

# spuštění vyhodnocení
def doExample():
    # načtení hodnot
    hodnoty = getValues()

```

```

        # kontrola vstupu
        chyba = self.kontrolaVstupu(hodnoty)

    # kontrola, zda byla zachycena chyba
    if chyba:
        # pokud chyba, nic se neprovede (chybový výstup je již
        # zobrazen)
        pass
    # pokud není chyba
    else:
        # hodnota ns se převede z procent
        hodnoty['ns'] = str(float(hodnoty['ns'])/100.)
        # předají se výsledné parametry k vyhodnocení
        self.vytvorVystup(nazev, hodnoty)
    pass

```

Nejprve se načtou hodnoty pomocí funkce *getValues* (bude uvedena dále), která je opět bez parametru, protože se navíc používá při funkci k uložení do souboru, kde je volána po stisku tlačítka. Poté se předané hodnoty vyhodnotí funkcí pro kontrolu vstupu (také bude uvedena dále v textu), a pokud jsou zadané vstupy v pořádku, do proměnné *chyba* se neuloží nic. V opačném případě předá příznak chyby. V dalším vyhodnocení se otestuje, zda byla chyba. Pokud ne, hodnota *ns* se převede z procent vydělením čísla konstantou 100 a hodnoty se spolu s názvem příkladu předají k vyhodnocení funkci *vytvorVystup*, která zajišťuje poslední fázi programu, tedy vyhodnocení dat a jejich následné zobrazení.

Pro úplnost je třeba uvést funkci pro získání hodnot z formulářů s názvem *getValues*, která je volána jak při vyhodnocení, tak při ukládání do souboru:

```

def getValues():
    values = {'S':Form_S.get().replace(",","."),\
              'nj':Form_nj.get().replace(",","."),\
              'T':Form_T.get().replace(",","."),\
              'c':Form_c.get().replace(",","."),\
              'ns':Form_ns.get().replace(",","."),\
              'dtvo':Form_tvo.get().replace(",","."),\
              'graf':Form_graf.get().replace(",","."),\
              'osmer':Form_osmer.get().replace(",","."),\
              'obsl':Form_obsl.get().replace(",",".")\
            }
    return(values)

```

Do slovníku *values* jsou vždy ukládány hodnoty, které jsou relevantní k danému příkladu a jsou pomocí funkce *get()* získány z formulářů, ať je jejich obsah jakýkoliv (platný, neplatný, prázdný). Navíc jsou před předáním hodnoty upraveny. Nastane-li situace, že obsahují jako oddělovač desetinné části čárku, je převedena na tečku, aby mohl být obsah vyhodnocen jako číslo. Python využívá pro oddělení desetinných míst právě tečku. Následně jsou hodnoty získané z formulářů vráceny pro další zpracování.

Navíc je v příkladu uvedena funkce pro zavolání uložení, které je již uvedeno mimo příklad. Používá se pro všechny příklady stejná funkce, pouze zavolání této funkce musí být v příkladu, protože reaguje na stisk tlačítka. Jednoduchá funkce je následující:

```

def uloz():
    hodnoty = getValues()
    self.ulozZadani(nazev,hodnoty,e01Window)

```

Do proměnné hodnoty se načtou hodnoty z formulářů. Dále se netestuje jejich chybovost, jsou uloženy tak, jak jsou (platné, neplatné, prázdné). Je zavolána funkce *ulozZadani*. Její kód je uveden výše. Předají se parametry s názvem příkladu, s hodnotami a z jakého okna byla funkce zavolána.

Jako předposlední příkaz v těle této funkce je uvedeno volání funkce sloužící k vytvoření menu v okně příkladu:

```
self.menuPříkladu(e01Window, uloz)
```

Jsou ji předány jako parametry okno, na které se aplikuje menu a název funkce, která bude spuštěna při stisknutí příslušné volby.

```
# menu pro uložení v každém příkladu
def menuPříkladu(self, window, uloz):
    # vytvoření objektu menu v příslušném okně
    e01Menu = Menu(window)

    # vytvoření podmenu
    soubor = Menu(e01Menu, tearoff=0)
    # přidání příkazu do podmenu
    soubor.add_command(label="Uložit", command=uloz)
    # vložení separátoru
    soubor.add_separator()
    # vložení dalšího příkazu
    soubor.add_command(label="Ukončit program", command=window.quit)
    # přidání podmenu pod hlavní menu
    e01Menu.add_cascade(label="Soubor", menu=soubor)

    # aplikace na okno
    window.config(menu=e01Menu)
```

Funkce převezme název okna a název funkce, kterou bude dále využívat. Pro okno vytvoří hlavní menu, které je následně využito pro vytvoření podmenu. Do tohoto podmenu je vložena první volba pomocí funkce *add_command*. Této volbě je přidána jako vykonatelný příkaz funkce, přesněji název funkce, která se má vykonat. Dále je vložen oddělovač pro větší přehlednost. Další příkaz je zařazen pod separátor a zajišťuje ukončení celého programu. Poté je podmenu začleněno do hlavního menu a hlavní menu je na závěr nastaveno pro zmíněné okno.

Jako další funkci je třeba zmínit kontrolu vstupu, která je potřebná při přebírání hodnot z formulářů. Její kód je následující:

```
def kontrolaVstupu(self, vstup):
    # pole pro uložení chybných hodnot
    chybnéHodnoty = []

    # postupné projití předaných hodnot
    for hodnota in vstup:
        # kontrola hodnot na číselný nebo nulový vstup
        if vstup[hodnota].replace(".", "").replace(",", "").isdigit()
        and vstup[hodnota].replace('0', '') != '':
            # pokud je v pořádku, neukládá se
            pass
        # pokud není v pořádku
        else:
            # speciální test pro hodnoty ze slevy A a slevy B
            if hodnota in
            ('i01', 'i02', 'i03', 'i04', 'i05', 'i06', 'i07', 'i08', 'i09', 'i10', \
```

```
'c01','c02','c03','c04','c05','c06','c07','c08','c09','c10')\
and vstup[hodnota] == '':
    # pokud je prázdná buňka, tak v pořádku
    pass
    # pokud chyba, tak se zapíše do pole
else:
    chybneHodnoty.append((hodnota, vstup[hodnota]))
```

Tato funkce je v podstatě určena pro zjištění chyb vstupů a jeho uložení do pole *chybneHodnoty*. Na začátku se pole vytvoří, aby se do něj mohly případně přidávat záznamy o chybách. Poté se postupně projdou hodnoty předané parametrem *vstup*, kde se jim v případě výskytu vymažou čárky a tečky. Je to z toho důvodu, že jsou hodnoty testovány, zda jsou číselné, pomocí funkce *isdigit()*, která posuzuje pouze číslice, nikoliv oddělovače. Za další zde nesmí být ani prázdný vstup, který by mohl způsobit havárii programu nebo nesprávný výsledek.

Dále je ve speciálním režimu sledován zápis hodnot z příkladů se slevou A a slevou B, kdy je možné některé parametry vynechat. Pokud hodnoty do této skupiny nepatří, jsou vyhodnoceny jako chybné. Stejně tak všechny ostatní, které neobsahují číselný vstup. Je-li hodnota chybná, zapíše se do pole, které je dále, po ukončení zpracování, vyhodnoceno:

```
# pokud je nějaká hodnota v poli
if chybneHodnoty:
    # nejprve začátek textu
    textChyby = "Chybně zadané hodnoty:\n"

    # postupné projití pole
    for hodnota in chybneHodnoty:
        # pokud existuje hodnota
        if hodnota[1]:

            # text chyby pro hodnotu
            textChyby += 'Hodnota ' + hodnota[1] + ' není platný vstup pro
' + hodnota[0] + '.\n'

        # jinak
        else:
            # text chyby bez hodnoty
            textChyby += 'Proměnná ' + hodnota[0] + ' nebyla zadána.'
+ '\n'

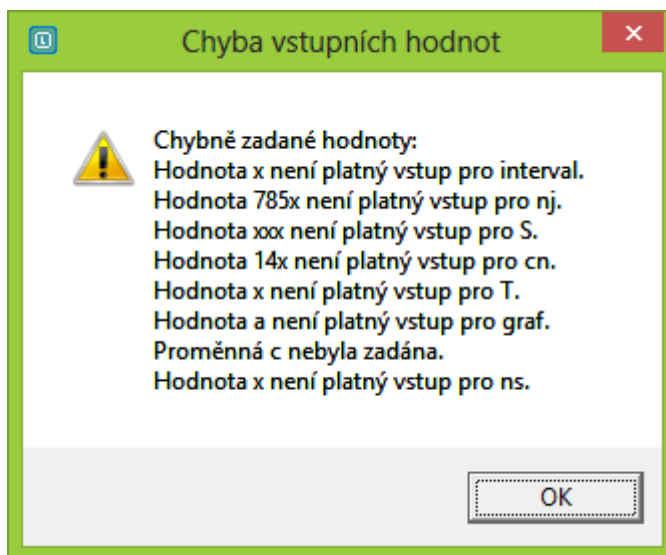
    # zobrazení informačního okna
    messagebox.showwarning("Chyba vstupních hodnot",textChyby)
    # vrátí True při chybě
    return True
else:
    # bez chyby vrátí False
    return False
```

Při splnění podmínky (vstup obsahuje chybné hodnoty) se postoupí k vytvoření chybové zprávy, která uživatele upozorní na chybný vstup. V prvním kroku se vytvoří proměnná, která je popsána chybovou hláškou a budou se do ní postupně přidávat řádky se všemi chybnými hodnotami.

V následujícím kroku se projdou všechny hodnoty, kdy se testuje, zda obsahují nepovolený obsah nebo jsou pouze prázdné. Jestliže obsahují neplatný text, je přidána hláška o neplatném vstupu. Pokud jsou prázdné, přidá se hláška o prázdném vstupu.

Po zpracování všech hodnot je vytvořeno informační okno k titulkem „Chyba vstupních hodnot“ a s vytvořeným textem.

Nastane-li situace, že byla chyba nalezena, funkce zobrazí informační okno a navíc vrátí hodnotu True, podle které lze identifikovat, že byla nalezena chyba a není možné pokračovat. V opačném případě je vrácena hodnota False a program může pokračovat.



Obr. 6 Ukázka hlášení při chybném vstupu

Poslední částí ukázkového příkladu je část, která spojuje výpočetní funkci a vykreslení výsledků v grafické formě a matematickém zápisu. Funkce má několik společných příkazů pro všechny příklady:

```
# vyhodnocení výsledků a vytvoření okna
def vytvorVystup(self, title, hodnoty):

    # vytvoření okna s výsledky
    window = pyplot.figure(figsize=(12, 8), facecolor="#CFCFCF")

    # nastavení titulku okna
    window.canvas.set_window_title(title)

    # vytvoření podtitulku
    pyplot.suptitle(title + ' - graf a výpočty', fontsize=13, fontweight='bold')

    # vyvolání funkce pro nastavení parametrů
    manager = pyplot.get_current_fig_manager()

    # nastavení ikony
    manager.window.wm_iconbitmap("favicon.ico")
```

Nejprve jsou předány parametry s názvem příkladu a hodnotami, které budou dále zpracovány. Funkce *figure* zajistí vytvoření hlavního okna. Nastaví se zde rozměry a barva pozadí. Následuje nastavení titulku okna podle názvu příkladu a také podtitulku, který se vloží na stránku s příkladem. Tato zdánlivá duplicita v názvu je proto, že v případě, že je příklad uložen do externího souboru, nebyl by již podle titulu identifikovatelný. Řešení vede k odstranění nejednoznačnosti.

Navíc je pro nastavení ikony použit stejný postup jako v předchozí případě při vytváření okna příkladu.

Dále už následují vždy data k jednotlivým případům. Kód k ostatním příkladům je vynechán a je zde uveřejněn pouze kód k P-systému:

```
# pokud je název příkladu
elif title == r'P - systém':
    # výpočet Qopt (informační)
    Qopt = formulas.q_optimal(hodnoty['S'], hodnoty['nj'], hodnoty['T'],
hodnoty['c'], hodnoty['ns'])

    # výpočet doby objednání (konečná hodnota)
    tc_year, tc_day = formulas.t(hodnoty['S'], Qopt, hodnoty['T'])
    # hodnota směrodatné odchylky pro P-systém
    sigmaT = formulas.sigmaT(hodnoty['dtvo'], tc_year, hodnoty['osmer'])
    # výpočet zvýšení pro dobu dodání
    u = formulas.signalni(hodnoty['S'], hodnoty['dtvo'])
    # výpočet navýšení
    more_w = formulas.navyseni(hodnoty['obsl'], sigmaT)

    # výpočet horní objednací hranice
    x_P = formulas.x_P(hodnoty['S'], hodnoty['dtvo'], tc_year, more_w)
```

Příklad je identifikován podle názvu uloženého v proměnné *title*. Následuje příslušný výpočet potřebných hodnot. Pro P-systém je to výpočet optimálního množství, které není konečné, pouze je použito pro další výpočet (P-systém nemá pevné objednací množství). Vypočtená perioda objednání už je konečná. Spočítá se upravená směrodatná odchylka s ohledem na dodací dobu a nutné zvýšení zásoby opět s ohledem na dodací dobu. Navýšení s přihlédnutím na směrodatnou odchylku P-systému a úroveň obsluhy je vypočítáno jako pojistná zásoba. V posledním kroku je dopočítána horní objednací úroveň, která je nejdůležitějším údajem pro příklad.

Program pokračuje generováním dat k příkladu, která jsou ukládána pro potřeby vykreslení grafického výstupu:

```
# vytvoření pole pro hodnoty do grafu
listSklad = []
# výchozí hodnota skladu
sklad = x_P
# čas dodání
time = int(hodnoty['dtvo'])
# čas objednání
time2 = int(tc_day)

# ušlá příležitost
usla_přilezitost_ks = 0;
usla_přilezitost = 0;

# příznaky pro odečítání času time a time2
priznak = False
priznak2 = False
```

Do pole se ukládá stav skladu v každém kroku. Graf tedy vykreslí po dnech stav zásob. Výchozí hodnota skladu je dána horní objednací hranicí. Je třeba nastavit čas objednání a čas dodání, které se následně použijí pro odpočítávání, kdy objednat a za jak dlouho objednávka dorazí na sklad. Dále jsou zde inicializovány proměnné pro ušlou příležitost, a to jak v kusech, ze kterých lze vyčíslit

ztrátu, tak v počtech, na kterých lze ověřit přesnost podle úrovně obsluhy. Nastavení příznaků je zde potřebné pro kontrolu odečítání dodacích a objednacích lhůt.

V každém okamžiku se tedy budou vyhodnocovat data, která byla v přechozím kroku získána výpočtem:

```
# generování hodnot pro graf
for i in range(1, (365*int(hodnoty['T'])+1)*int(hodnoty['graf'])):
    # přidání hodnoty
    listSklad.append(sklad)

    # testování, zda už je čas objednat
    if time2 < 1:
        # znovu nastavit čas objednání
        time2 = int(tc_day)
        # správně nastavit příznaky
        priznak2 = True
        priznak = True

    # postupné odčítání času
    time2 -= 1

    # pokud je priznak, tzn. je čas objednat
    # nastavení správného množství objednání
    if priznak:
        objednavka = x_P - sklad
        priznak = False

    # pokud je priznak2, tzn. je objednáno
    if priznak2:
        # pokud je čas dodání
        if time < 1:
            # správné nastavení příznaku
            priznak2 = False

            # objednávka dorazila na sklad
            sklad += objednavka
            # obnoví čas dodávky
            time = int(hodnoty['dtvo'])

        # odčítá čas dodávky
        time -= 1
```

Délka generovaných dat se odvozuje od násobku T , který je zadáván v rocích, a proto je třeba, aby byla hodnota vynásobena 365 (přibližný počet dní v roce). Uživatel si násobek může zvolit. V prvním kroku je vždy přidán stav skladu.

Pak se testuje příznak, zda se má již objednat. Pokud ano, nastaví se příznak pro zahájení objednání a uloží se hodnoty pro objednání. Obnoví se čas pro počítání do dalšího objednání. Pokud ne, odečítá se jednička, symbolizující jeden uběhlý den. Jestliže je nastaven *priznak*, uloží se velikost objednávky a *priznak* se odnastaví, aby byla velikost objednávky zachována až do objednání.

Je-li je nastaven *priznak2*, odečítá se dodací lhůta. Nastane-li čas dodání, odnastaví se *priznak2*, protože se právě dodává a hodnota skladu se navýší o velikost objednávky. Zároveň se nastaví čas dodání na výchozí hodnotu. Když ne, odečítá se jednička, symbolizující jeden uběhlý den.

V dalším je popsán konec cyklu:

```
# spotřeba je počítána pomocí normálního rozdělení
spotreba = int(float(numpy.random.normal(u, float(hodnoty['osmer']),
1)[0]) /float(hodnoty['dtvo']))

# pokud by byla záporná poptávka
if spotreba < 0:
    # nastaví se nula
    spotreba = 0

# pokud by byla poptávka větší než zásoba
if sklad - spotreba < 0:
    # počítají se kusy a výskyt
    usla_prilezitost_ks += sklad - spotreba
    # nastaví se nula
    sklad = 0

else:
    sklad -= spotreba
```

Spotřeba je vypočítána ze zadaných dat a vychází z normálního rozdělení. Pro zajištění špatného výpočtu je doplněna podmínka, která se aktivuje při záporné spotřebě (což není možné v reálné situaci, rozdělení však toto umožňuje) a spotřeba se nastaví na nulu. Dále se nastaví hodnoty ušlé příležitosti, jestliže je zásoba na skladě menší než aktuální spotřeba. Sklad se nastaví na nulu, protože hodnota skladu nemůže být záporná. Vyskytuje-li se na skladě dostatečná zásoba, odečte se spotřeba ze skladových zásob.

Tímto je dokončeno generování dat.

Data je třeba ještě prezentovat, a to v podobě grafu i v matematickém zápise. Nejprve se vytvoří graf, jehož tvorba a zobrazení je jednoduché:

```
# nastavení plochy
pyplot.axes([0.08, 0.48, 0.84, 0.47], axisbg="white", frameon=True)
# nastavení hodnot x a y
pyplot.plot(range(len(listSklad)), listSklad, 'y-')
# nastavení minimální hodnoty y
pyplot.ylim(ymin=0)
# zobrazení mřížky
pyplot.grid(True)
```

Nejprve je nastavena plocha grafu a to do horní oblasti. Rámeček je viditelný a barva osy je bílá. Poté jsou do grafu zaneseny hodnoty, přičemž pro osu x je to počet záznamů, symbolizující počet dnů v generovaných datech. Pro hodnotu y jsou to stavy skladu. Pro vykreslení je použita žlutá plná čára. Z důvodu lepší orientace, aby bylo vidět, že se skladové zásoby blíží nule, je minimum grafu nastaveno na nulu. Tedy vždy zobrazí nulu. Poté následuje nastavení mřížky.

Druhou vykreslenou plochou je plocha s matematickým zápisem:

```
# nastavení plochy
pyplot.axes([0.02, 0.02, 0.96, 0.4], axisbg="white", frameon=True)
# nastavení rozsahu x
pyplot.gca().set_xlim(0., 1.)
# nastavení rozsahu y
pyplot.gca().set_ylim(0., 0.14)
# vypnutí hodnot x a y na ose
pyplot.gca().set_xticklabels("", visible=False)
pyplot.gca().set_yticklabels("", visible=False)
```

Zde se nastaví plocha do dolní oblasti s bílými osami a zapnutým rámečkem. Rozsah okna je nastaven pro určení matematických vzorců, který bude následovat dále. Hodnoty na ose jsou vypnuté, protože plocha slouží pro zapsání matematických hodnot a rozsah tedy nepotřebuje.

Jako poslední je uveden kód, pomocí kterého se zobrazují vzorce, a to jak v zápise s proměnnými, tak s čísly a výsledky:

```
# vzorec pro Qopt
text = r'$Q_{opt} \quad \backslash=\backslash \quad \sqrt{\frac{2S_{nj}}{T_{n_{sc}}}} \quad \backslash=\backslash \quad \sqrt{\frac{2}{\backslash\cdot\backslash}} + \quad self.cisloDoZapisu(hodnoty['S']) \quad + \quad '\backslash\cdot\backslash' + \quad self.cisloDoZapisu(hodnoty['nj']) \quad + \quad '{}{}' + \quad self.cisloDoZapisu(hodnoty['T']) + \quad '\backslash\cdot\backslash' + \quad self.cisloDoZapisu(hodnoty['ns']) + \quad '\backslash\cdot\backslash' + \quad self.cisloDoZapisu(hodnoty['c']) + \quad '{}{} \quad \backslash=\backslash' + \quad str(Qopt) + \quad '\backslash\$ks'

pyplot.text(0.04, 0.11, text, fontsize=16)

# vzorec pro tc
text = r'$t_c \quad \backslash=\backslash \quad \frac{Q}{(ST)} \quad \backslash=\backslash \quad \frac{\quad}{\quad} + \quad self.cisloDoZapisu(Qopt) + \quad r'\{(\quad + \quad self.cisloDoZapisu(hodnoty['S']) \quad + \quad r'\backslash\cdot\backslash' + \quad self.cisloDoZapisu(hodnoty['T']) + \quad r')\} \quad \backslash=\backslash \quad ' + \quad self.cisloDoZapisu(tc\_year, 5) + \quad r' \backslash\$' + \quad r'roku' + \quad r'\$ \backslash\cdot\backslash\cdot\backslash\cdot\backslash' + \quad self.cisloDoZapisu(tc\_day) + \quad r'\backslash\$' + \quad r'dní'

pyplot.text(0.04, 0.085, text, fontsize=16)

... (zkrácení kódu) ...
```

Kód je bohužel ve svém zápise špatně čitelný. Provede se přiřazení textu do proměnné, která je následně vykreslena na určité místo v určené velikosti. Z kódu je patrné použití funkce *self.cisloDoZapisu*, která se používá pro úpravu hodnot před zápisem. Kód funkce je uveden výše.

Jako poslední je nutné celý graf zobrazit, což se provádí příkazem, který je zároveň poslední v celé funkci:

```
# zobrazení výsledku
pyplot.show()
```


8 ZÁVĚR

Práce klasifikuje systémy řízení zásob, a to standardním způsobem, který sleduje dělení na modely deterministické a stochastické. Během postupného procházení jednotlivých deterministických modelů je patrné, že jsou v některých případech základem pro modely stochastické.

Uvedením příkladů každého systému bylo možné usnadnit vývoj programu. Při implementaci, která probíhala v programovacím jazyce Python, bylo těchto příkladů využito pro analýzu jednotlivých modelů a zároveň také k testování, zda jsou výsledné výpočty správné.

Při ručním počítání je běžné, že se některé výpočty zaokrouhlují k číslům, která dávají větší smysl. Například, jestli je termín objednání jednou za 13 dní, je přiměřené, aby se tento termín o den posunul a k němu se poté dopočítaly patřičné hodnoty. Program ovšem toto nereflektuje, počítá vždy s téměř přesnými hodnotami. Proto rozhodně není ideální pomůckou, ovšem při správném použití může usnadnit výpočet a přinést rychlou představu o průběhu toku skladových zásob. To může být pomoc při rozhodování, jak navrhnout systém zásob.

Systém je jednak uložen ve zdrojovém kódu jazyka Python, a také je pomocí knihovny třetích stran převeden do spustitelné podoby na systému Windows. Při komplementaci do spustitelného balíčku jsou přibaleny všechny potřebné knihovny a software je tak možné spustit bez jakékoliv nadstandardní instalace softwaru.

Program byl navržen přesně podle uvedených příkladů. Menu příkladu je tedy rozděleno na deterministické a stochastické modely. Každý příklad má svůj specifický typ okna pro zadávání příkladů. Příklady je možné ukládat a posléze načítat. Všechny příklady uvedené v práci jsou uloženy, aby je bylo možné rovnou načítat a spouštět.

Výstup má jednotnou podobu, na které je vidět graf a přidružené výpočty. To je výhodné, protože jsou všechny výsledky souhrnně na jedné stránce, kterou je navíc možné uložit.

Implementace systémů řízení zásob byla dokončena podle předchozí klasifikace. Každý zde uvedený příklad se podařilo algoritmizovat, řešení využít a kompletovat do jednotného programu. V práci je popsán téměř kompletní vývoj programu. Jsou zde uvedeny všechny nezbytné i podpůrné funkce, a také kód ukázkového příkladu, pro který byl zvolen P-systém. Zdrojový kód je kompletně okomentován a vysvětlen.

Program je tedy v rozsahu zadání kompletní. Do programu je však nutné zadávat adekvátní čísla, není zcela ošetřen pro nesprávný vstup. Je schopen rozpoznat číselné hodnoty, které jako jediné přijímá. Pokud jsou mu ovšem zadána nesmyslná nebo chybná data, neumí tento vstup rozeznat a pokusí se s nimi počítat. Výstup je pak nesmyslný a není možné z něj vyvozovat závěry.

9 POUŽITÁ LITERATURA

- [1] HUŠEK, Roman a Josef LAUBER. Aplikace stochastických procesů II. Praha: SPN Praha, 1982.
- [2] GROS, Ivan. Matematické modely pro manažerské rozhodování. Praha: Vydavatelství VŠCHT, 2009, 282 s. ISBN 978-80-7080-709-5.
- [3] RÁLEK, Petr, Josef NOVÁK a Josef CHUDOBA. TECHNICKÁ UNIVERZITA V LIBERCI. *Metody užívané v logistice* [online]. 2010, 82 s. [cit. 2015-05-07]. Dostupné také z: http://www.nti.tul.cz/cz/images/0/0a/Mul_skripta_101101.pdf
- [4] KINSHOOK, Chaturvedi. Inventory management. In: *Slideshare* [online]. 2011 [cit. 2015-05-08]. Dostupné z: <http://www.slideshare.net/kinshookc/inventory-control-8025119>
- [5] SIXTA, Josef. *Logistika: teorie a praxe*. Vyd. 1. Brno: CP Books, 2005, 315 s. Praxe manažera (CP Books). ISBN 80-251-0573-3.
- [6] Bartmann, D. and Beckmann, M. J.: Inventory Control. Models and Methods. Springer-Verlag, Berlin, 1992.
- [7] Jablonský, J.: Operační výzkum. Vysoká škola ekonomická, Fakulta informatiky a statistiky, Praha, 2001.

SEZNAMY

Seznam grafů

Graf 1 Příklad běžné a pojistné zásoby	16
Graf 2 Časový průběh velikosti zásob při modelu optimální velikosti objednávky.....	19
Graf 3 Velikost nákladových křivek vzhledem k obj. množství u slevy typu A.....	23
Graf 4 Q-systém	28
Graf 5 P-systém.....	31

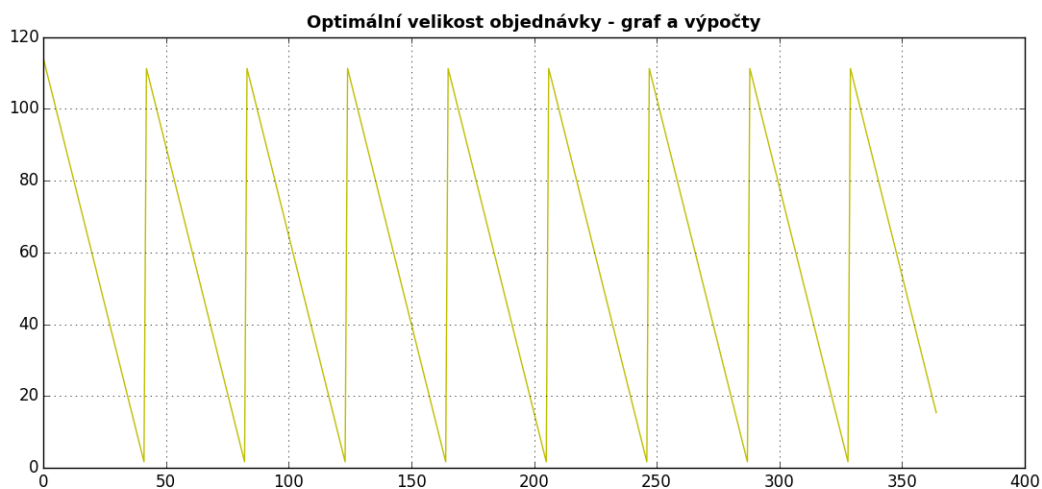
Seznam tabulek

Tab. 1 Cena v jednotlivých intervalech.....	21
Tab. 2 Vstupní data příkladu s optimální velikostí objednávky	33
Tab. 3 Výsledky příkladu s optimální velikostí objednávky	34
Tab. 4 Vstupní data příkladu s optimální velikostí objednávky při nespojitém množství ...	34
Tab. 5 Výsledky příkladu s optimální velikostí objednávky při nespojitém množství	35
Tab. 6 Vstupní data příkladu s opt. velikostí obj. při nespojitém objednacím množství	36
Tab. 7 Výpočetní tabulka pro nespojité objednací množství	36
Tab. 8 Výsledek příkladu s opt. velikostí obj. při nespojitém objednacím množství.....	36
Tab. 9 Ceník k příkladu se slevou A	37
Tab. 10 Vstupní data k příkladu se slevou A	37
Tab. 11 Výpočetní tabulka k příkladu se slevou A	38
Tab. 12 Výsledky k příkladu se slevou A	38
Tab. 13 Ceník k příkladu se slevou B	39
Tab. 14 Vstupní data k příkladu se slevou B.....	39
Tab. 15 Výpočetní tabulka k příkladu se slevou B.....	40
Tab. 16 Výsledky k příkladu se slevou A	40
Tab. 17 Vstupní data k příkladu Q-systém.....	41
Tab. 18 Výsledku k příkladu Q-systém.....	42
Tab. 19 Vstupní data k příkladu P-systém	42
Tab. 20 Výsledky k příkladu P-systém	43

Seznam obrázků

Obr. 1 Schéma programu	47
Obr. 2 Menu hlavního programu.....	48
Obr. 3 Ukázkový vstup programu	49
Obr. 4 Ukázkový výstup programu.....	50
Obr. 5 Vyplněný vstup příkladu P-systém	55
Obr. 6 Ukázka hlášení při chybném vstupu	63

PŘÍLOHY

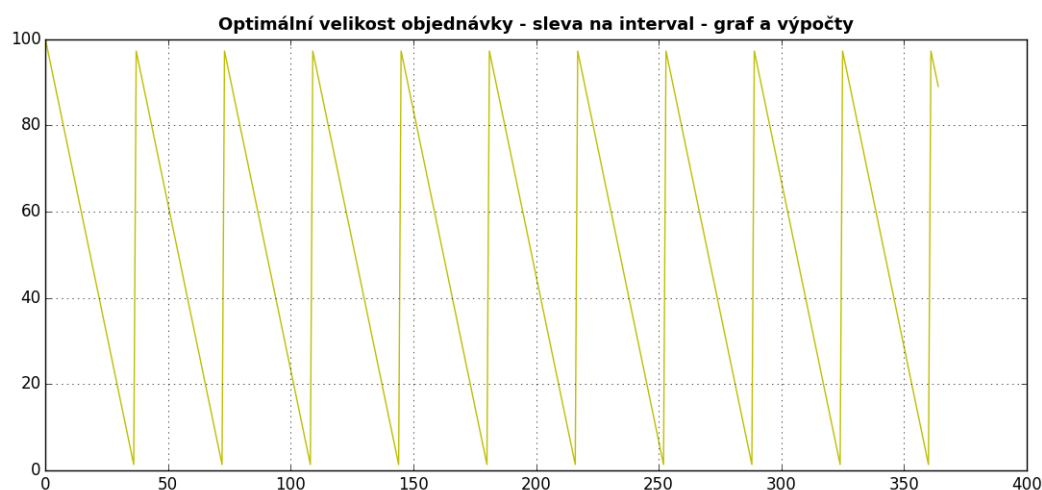


$$Q_{opt} = \sqrt{\frac{2Sn_j}{Tn_s c}} = \sqrt{\frac{2 \cdot 1\,000 \cdot 7\,500}{1 \cdot 0,18 \cdot 6\,500}} = 114 \text{ ks}$$

$$N(Q_{opt}) = \sqrt{2STcn_s n_j} = \sqrt{2 \cdot 1\,000 \cdot 1 \cdot 6\,500 \cdot 0,18 \cdot 7\,500} = 132\,477 \text{ Kč}$$

$$t_c = \sqrt{\frac{2Tn_j}{Sn_s c}} = \sqrt{\frac{2 \cdot 1 \cdot 7\,500}{1\,000 \cdot 0,18 \cdot 6\,500}} = 0,11322 \text{ roku} \doteq 41 \text{ dní}$$

Příloha 1 Výsledek 1. Příkladu



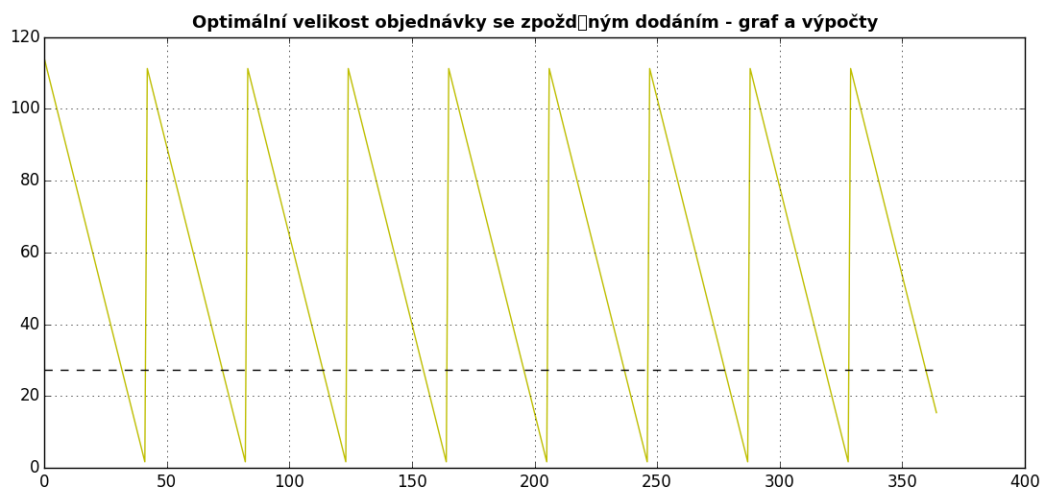
$$Q_{opt} = \sqrt{\frac{2Sn_j}{Tn_s c}} = \sqrt{\frac{2 \cdot 1\,000 \cdot 7\,500}{1 \cdot 0,18 \cdot 6\,500}} = 114 \text{ ks}$$

$$Q = 100 \text{ ks}$$

$$N(Q) = \frac{Q}{2}Tn_s c + \frac{S}{Q}n_j = \frac{100}{2} \cdot 1 \cdot 0,18 \cdot 6\,200 + \frac{1\,000}{100}7\,500 = 130\,800 \text{ Kč}$$

$$t_c = \frac{TQ}{S} = \frac{1 \cdot 100}{1\,000} = 0,1 \text{ roku} \doteq 36 \text{ dní}$$

Příloha 2 Výsledek 2. příkladu



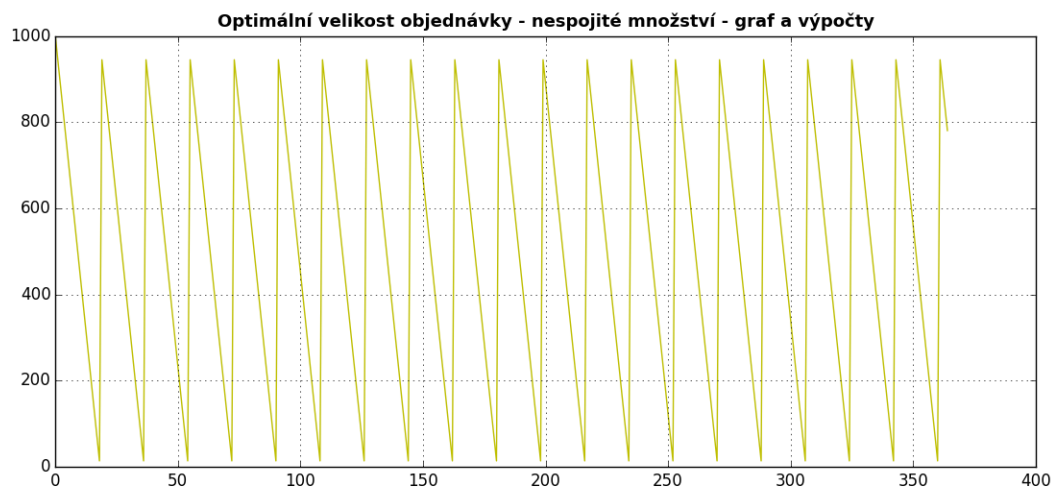
$$Q_{opt} = \sqrt{\frac{2Sn_j}{Tn_sc}} = \sqrt{\frac{2 \cdot 1\,000 \cdot 7\,500}{1 \cdot 0,18 \cdot 6\,500}} = 114 \text{ ks} \quad \text{Kontinuální objednávky} = 0$$

$$\text{Signální stav zásob } x_d = St_{vo} \bmod Q_{opt} = 1\,000 \cdot \frac{10}{365} \bmod Q_{opt} = 27 \text{ ks}$$

$$N(Q_{opt}) = \sqrt{2STcn_s n_j} = \sqrt{2 \cdot 1\,000 \cdot 1 \cdot 6\,500 \cdot 0,18 \cdot 7\,500} = 132\,477 \text{ Kč}$$

$$t_c = \sqrt{\frac{2Tn_j}{Sn_sc}} = \sqrt{\frac{2 \cdot 1 \cdot 7\,500}{1\,000 \cdot 0,18 \cdot 6\,500}} = 0,11322 \text{ roku} \doteq 41 \text{ dní}$$

Příloha 3 Výsledek 3. příkladu



$$Q_{opt} = \sqrt{\frac{2Sn_j}{Tn_sc}} = \sqrt{\frac{2 \cdot 20\,000 \cdot 10\,500}{1 \cdot 0,17 \cdot 3\,200}} = 1\,000 \text{ ks}$$

$$N(Q) = \frac{Q}{2}Tn_sc + \frac{S}{Q}n_j = \frac{1\,000}{2} \cdot 1 \cdot 0,17 \cdot 3\,200 + \frac{20\,000}{1\,000} \cdot 10\,500 = 482\,000 \text{ Kč}$$

$$t_c = \frac{Q}{S} = \frac{1\,000}{20\,000} = 0,05 \text{ roku} \doteq 18 \text{ dní}$$

Příloha 4 Výsledek 4. příkladu



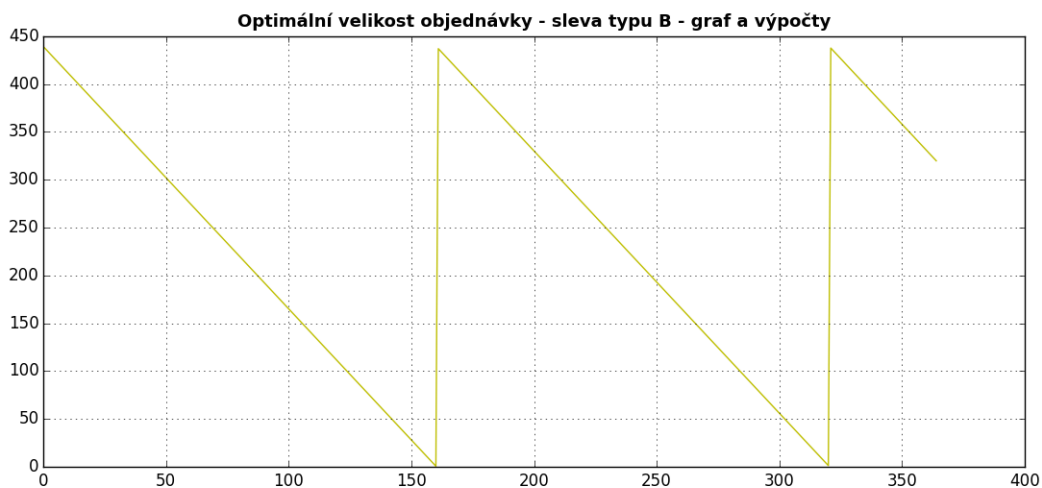
$$Q_{opt} = \sqrt{\frac{2Sn_j}{Tn_sc}} = \sqrt{\frac{2 \cdot 25\,000 \cdot 5\,500}{1 \cdot 0,16 \cdot 2\,400}} = 830 \text{ ks}$$

$$Q = 900 \text{ ks}$$

$$N(Q) = \frac{Q}{2}Tn_sc + \frac{S}{Q}n_j = \frac{900}{2} \cdot 1 \cdot 0,16 \cdot 2\,400 + \frac{25\,000}{900} \cdot 5\,500 = 325\,578 \text{ Kč}$$

$$t_c = \frac{QT}{S} = \frac{900 \cdot 1}{25\,000} = 0,0332 \text{ roku} \doteq 12 \text{ dní}$$

Příloha 5 Výsledek 5. příkladu



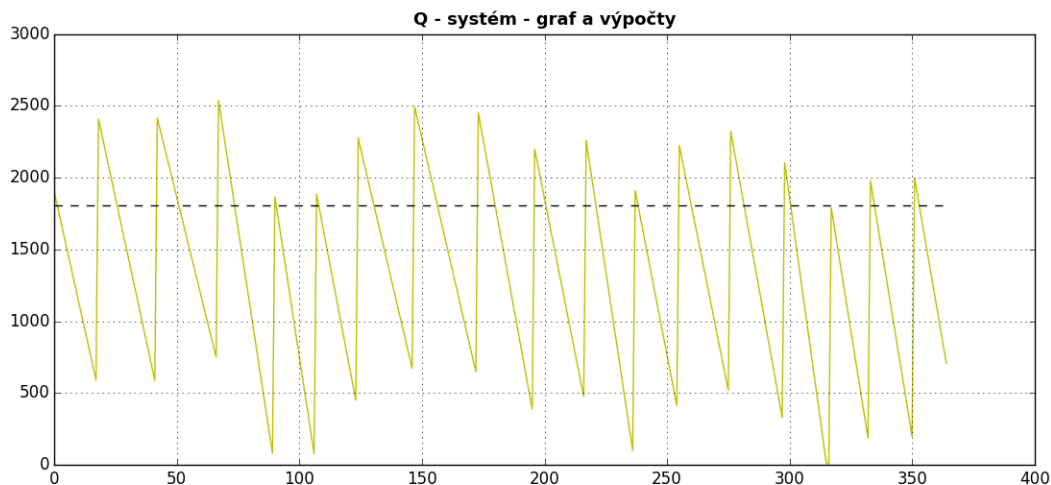
$$Q_{opt} = \sqrt{\frac{2Sn_j}{Tn_sc}} = \sqrt{\frac{2 \cdot 1\,000 \cdot 250}{1 \cdot 0,13 \cdot 198}} = 439 \text{ ks}$$

$$N(Q) = Sc_i + \frac{1}{2}Tn_s(F_{i-1} - Q_dc_i) + \sqrt{2STn_sA_i}$$

$$N(Q) = 1\,000 \cdot 198 + \frac{1}{2} \cdot 1 \cdot 0,13(62\,250 - 300 \cdot 198) + \sqrt{2 \cdot 1\,000 \cdot 1 \cdot 0,13 \cdot 2\,500} = 198\,992 \text{ Kč}$$

$$t_c = \frac{Q}{(ST)} = \frac{439}{(1\,000 \cdot 1)} = 0,439 \text{ roku} \doteq 160 \text{ dní}$$

Příloha 6 Výsledek 6. příkladu



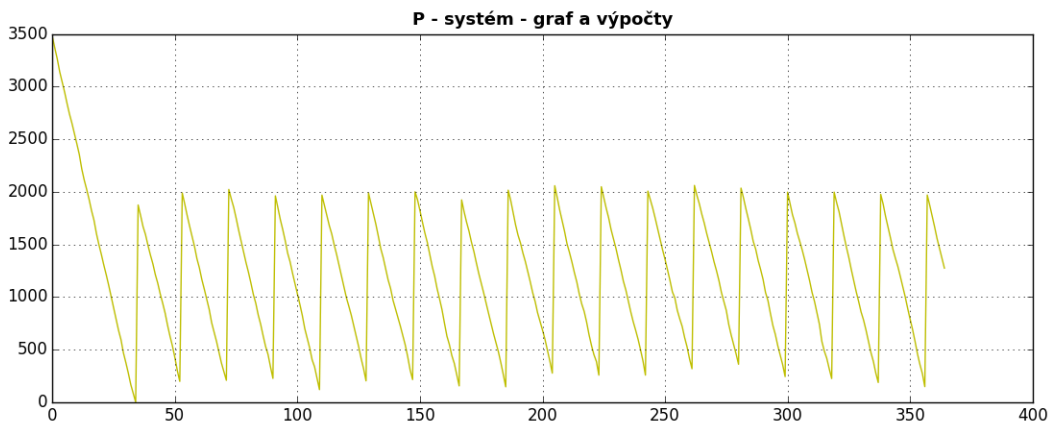
$$Q_{opt} = \sqrt{\frac{2Sn_j}{Tn_{sc}}} = \sqrt{\frac{2 \cdot 36\,000 \cdot 1\,200}{1 \cdot 0,2 \cdot 120}} = 1\,898 \text{ ks}$$

$$N(Q_{opt}) = \sqrt{2STcn_s n_j} = \sqrt{2 \cdot 36\,000 \cdot 1 \cdot 120 \cdot 0,2 \cdot 1\,200} = 45\,537 \text{ Kč}$$

$$w \geq z^* \sigma_{Q_d} = 1,64485 \cdot 200 = 329 \text{ ks}$$

$$r^* + w = 1\,479 + 329 = 1\,808 \text{ ks}$$

Příloha 7 Výsledek 7. příkladu



$$Q_{opt} = \sqrt{\frac{2Sn_j}{Tn_{sc}}} = \sqrt{\frac{2 \cdot 36\,000 \cdot 1\,200}{1 \cdot 0 \cdot 120}} = 1898 \text{ ks}$$

$$t_c = \frac{Q}{(ST)} = \frac{1\,898}{(36\,000 \cdot 1)} = 0,05272 \text{ roku} \doteq 19 \text{ dní}$$

$$\sigma_{t_{vo}+t_c} = \sqrt{(t_{vo}+t_c) \cdot \sigma_{Q_d}^2} = \sqrt{\left(\frac{15}{365} + 0,05272\right) 200^2} = 61 \text{ ks}$$

$$w \geq z^* \sigma_{Q_d} = 1,64485 \cdot 61 = 101 \text{ ks}$$

$$x_h = S(t_{vo}+t_c) + w = 36\,000 \left(15 + \frac{19}{365}\right) + 101 = 3\,478 \text{ ks}$$

Příloha 8 Výsledek 8. příkladu