ChatGPT

Analyse du planificateur Bühlmann ZH-L16C avec Gradient Factors (GF)

Implémentation du modèle ZH-L16C dans le dépôt

Le dépôt `buhlmann-planner-agent` contient une implémentation en TypeScript de l'algorithme de décompression Bühlmann ZH-L16C avec Gradient Factors. Le code se structure en plusieurs modules clairs :

- **Constantes du modèle :** Le fichier `core/constants.ts` définit les 16 compartiments tissulaires du modèle ZH-L16C (demi-vies pour N_2 et He), ainsi que les coefficients *a* et *b* de Bühlmann pour l'azote et l'hélium [1]. Ces valeurs correspondent aux données publiées pour ZH-L16C : par exemple, le compartiment le plus rapide (N_2 5 min) a *a=1.1696* et *b=0.5578*, et le plus lent (N_2 635 min) a *a=0.2327* et *b=0.9653* [2]. Les coefficients pour l'hélium sont également fournis (par ex. 1,88 min avec *a=1.6189*, *b=0.4770* pour He) [1]. *Ces constantes sont conformes aux tables Bühlmann ZH-L16C de référence* (les différences éventuelles sont minimes, ex. *a*=0,4701 vs 0,4710 sur un compartiment, sans impact significatif).

- **Modèle de tissus et gaz :** Le code représente l'état des tissus par deux tableaux de pression partielle (pN2 et pHe pour chaque compartiment) [3]. Initialement, à la surface, chaque compartiment est saturé à l'air (N_2 ~0,79 bar, He ~0) en tenant compte de la pression eau-vapeur dans les poumons ($PH_2O \approx 0{,}0627$ bar) [3]. Le gaz respiratoire est décrit par les fractions FO_2, FHe, FN_2 (calculées pour s'assurer que FO_2+FHe+FN_2=1) [4]. *L'inclusion de l'hélium permet de planifier des mélanges Trimix, même si le planificateur ne gère qu'un seul gaz pour l'instant.*

- **Calcul de la saturation/désaturation :** La fonction `updateConstantDepth` met à jour les pressions inertielles des tissus lorsqu'on passe un certain temps à une profondeur constante [5]. Elle utilise l'équation exponentielle de Haldane (forme de Schreiner) pour chaque compartiment :

- $k = \ln(2)/T_{1/2}$ est la constante de temps du compartiment.
- $P_{amb} = P_{\text{surface}} + 0,1 \times \text{profondeur (m)}$ convertit la profondeur en pression absolue [6].
- $P_{insp} = (P_{amb} - PH_2O) \times f_{inert}$ est la pression partielle inspirée d'un gaz inerte (N_2 ou He) dans les poumons [6].
- La mise à jour est : $P_{\text{tissuel,new}} = P_{\text{tissuel,old}} + (P_{insp} - P_{\text{tissuel,old}}) \times (1 - e^{-k\Delta t})$ [7], pour un intervalle de Δt minutes.

Cette formule est conforme à l'équation de Schreiner/Haldane utilisée dans la littérature [8] [9]. Elle s'applique séparément à N_2 et He, assurant le suivi correct des deux gaz inertes dans chaque compartiment.

- **Formule de plafond (M-value) avec Gradient Factors :** L'algorithme calcule le **plafond de décompression** (profondeur minimale à laquelle on peut monter sans dépasser les limites) pour chaque compartiment via la formule d'Erik Baker intégrant les GF. Dans le code, la fonction `ceilingForComp` implémente :

$$P_{amb,min} = \frac{P_{tiss} - GF \cdot a}{\frac{GF}{b} + (1 - GF)}$$

où $P_{tiss} = P_{N2}+P_{He}$ est la pression totale inerte dans le compartiment, et a,b les coefficients Bühlmann interpolés pour ce mélange [10] [11]. Le résultat $P_{amb,min}$ est converti en *profondeur de plafond* en mètres (en soustrayant la pression surface puis divisant par 0,1 bar/m) [12]. *Cette formule correspond exactement à celle décrite par Erik Baker pour appliquer un gradient factor au modèle de Bühlmann* [13] [11]. Elle revient à réduire proportionnellement l'écart entre la ligne de surpression maximum (ligne M-value) et la pression ambiante, afin d'imposer une marge de sécurité : un GF de 100% reproduit la limite Bühlmann classique, tandis qu'un GF plus bas exige un plafond plus profond (donc plus conservateur) [14] [15].

À noter : Pour combiner azote et hélium, le code interpole les coefficients a et b en pondérant par la part de chaque gaz dans le compartiment [16] [17]. C'est cohérent avec la formule standard où $a = \frac{a_{N2}P_{N2}+a_{He}P_{He}}{P_{N2}+P_{He}}$ et de même pour b [17]. Ainsi, chaque compartiment a son propre a,b effectif dépendant de la proportion de N_2 vs He qu'il contient, ce qui est conforme aux modèles Bühlmann multi-gaz.

Calcul de la décompression et enchaînement des paliers

La fonction principale `planDecompression(depth, bottomTime, gas, gfLow, gfHigh, opts)` génère le plan de plongée décompressive [18]. Voici le déroulement logique :

1. **Initialisation et descente :** On initialise les tissus à saturation surface (air) [3]. Puis, si une profondeur >0 est visée, la descente est simulée minute par minute à un taux fixe (ici ~19 m/min) [19]. Chaque minute, on calcule la nouvelle profondeur atteinte et on met à jour les tissus avec `updateConstantDepth` sur 1 minute [20]. Cette simulation pas-à-pas vise à mieux coller à la réalité de l'absorption inerte pendant la descente (plutôt qu'un calcul instantané). *Le temps de descente s'accumule dans le compteur TTS (Time To Surface) du plan* [19].

2. **Temps au fond (Bottom time) :** Le séjour au fond à profondeur maximale est appliqué en une étape : on met à jour les tissus sur `bottomMin` minutes à profondeur max [21]. Cela utilise l'équation exponentielle, ce qui équivaut à intégrer le chargement inertiel sur toute la durée. Le temps fond s'ajoute également au TTS pour l'instant [22].

3. **Calcul du premier** plafond **et du premier palier :** À l'issue du fond, on calcule la profondeur de plafond la plus contraignante en appliquant le Gradient Factor bas (GF_low) sur tous les compartiments [23]. On obtient ainsi la **profondeur du premier palier requis**. Le code arrondit ce plafond au palier de 3 m immédiatement supérieur (plus profond) et le contraint à être au moins égal au dernier palier choisi par l'utilisateur (3 m par défaut, ou 6 m si option « dernier palier 6 m » activée) [23]. Par exemple, si le calcul donne un plafond à 7,2 m, le premier palier sera fixé à 9 m. S'il donne 2 m mais que l'option dernier palier 3 m est activée, on prendra 3 m minimum. Ce palier initial est stocké dans `firstStop` [23].

4. **Remontée vers le premier palier :** Le programme simule ensuite la **remontée du fond vers la profondeur du premier palier**. Cela se fait, comme la descente, par pas d'une minute à un taux de remontée fixe (ici ~9 m/min) [24]. À chaque minute, la nouvelle profondeur est calculée (on ne dépasse pas la profondeur du palier) et les tissus sont mis à jour sur 1 min à cette nouvelle profondeur [25]. On incrémente le TTS d'une minute par pas. Cette phase s'arrête une fois la profondeur du palier atteinte. *Remarque :* ce procédé discret peut parfois légèrement surestimer

le temps : si, par exemple, 2,5 minutes suffiraient, le code prendra 3 minutes en faisant un dernier palier de 0,5 m sur 1 min complète [26]. Cela ajoute une petite marge conservatrice sur le temps de remontée.

5. **Boucle de paliers successifs :** La partie principale de l'algorithme est une boucle qui gère tous les paliers de décompression, de la profondeur du premier palier jusqu'au dernier palier (3 ou 6 m) [27] [28]. À chaque palier courant (variable `stopDepth`), on procède ainsi :

6. On initialise un compteur `held = 0` minute pour le temps passé à ce palier [29].
7. **Calcul du prochain palier et du GF intermédiaire :** On calcule la profondeur du **palier suivant** envisagé, qui est 3 m plus haut (sauf si le palier courant est le dernier palier, alors le suivant serait la surface) [30]. On détermine la valeur de GF **à la profondeur du palier suivant**, notée `gfNext` [30]. En effet, le GF appliqué augmente linéairement au fur et à mesure qu'on se rapproche de la surface. Le code réalise l'interpolation linéaire entre GF_low (au premier palier) et GF_high (à la surface) en fonction de la profondeur courante : plus on est proche de la surface, plus le GF effectif se rapproche de GF_high [31]. *Cette interpolation correspond à la recommandation d'Erik Baker d'augmenter graduellement le facteur de gradient pendant la remontée, depuis GF_low au premier palier jusqu'à GF_high en surface* [32] [15]. Concrètement, si firstStop=30 m, GF_low=30% et GF_high=80%, alors à mi-profondeur (~15 m) le GF effectif sera ~55%, etc.

8. **Vérification du critère de départ du palier :** À l'aide de `gfNext`, on recalcule le plafond *actualisé* des tissus (`ceilNext`) tel qu'il serait en montant au palier suivant [30]. Tant que ce plafond calculé `ceilNext` est **au-dessus** du palier suivant, le plongeur **ne peut pas quitter** le palier actuel sans risque [33]. Autrement dit, on doit rester au palier courant jusqu'à ce que la sursaturation diminue suffisamment pour que le palier suivant devienne autorisé. Dans le code, la condition de sortie est :

   `canLeave` **= (** `ceilNext` ≤ `nextDepth` **) ET (si palier final,** `held` ≥ `minLast` **)** [33].

   On exige en plus qu'au dernier palier (`stopDepth == lastStopDepth`), le temps `held` ait atteint la durée minimale de dernier palier voulue (option `minLastStopMinutes`, par ex. 1 min) [33]. Cela garantit un palier final de sécurité d'au moins X minutes même si le calcul pur indiquerait qu'on peut remonter plus tôt. - **Si on ne peut pas encore quitter le palier :** on simule une minute de plus à cette profondeur en appelant `updateConstantDepth` pour 1 min [34]. Les pressions inertes diminuent un peu dans les tissus (on "désature"). On incrémente `held` et TTS de 1 min [34], puis on recompute `ceilNext` au nouvel état tissulaire pour voir si la condition de départ est satisfaite. Ce loop interne représente l'attente au palier jusqu'à ce que les tissus "permettent" de monter au palier suivant. - **Validation du palier atteint :** Si l'on a passé du temps (`held > 0`) à ce palier, on enregistre ce palier dans la liste des stops avec sa profondeur, le temps passé et le gradient factor effectif à cette profondeur [28]. (Le GF affiché permet de voir à quel pourcentage de la marge on a effectué ce palier, ce qui est un détail informatif). - **Transition au palier suivant :** On calcule ensuite la profondeur du palier suivant (`nextDepth = stopDepth - 3m`) ou surface si on était à 3 m [35]. On simule la remontée vers ce palier suivant de la même manière qu'avant : par pas d'une minute à 9 m/min [36], en mettant à jour les tissus progressivement. Ce mouvement ajoute encore du temps au TTS. Une fois arrivé à `nextDepth`, ce devient le nouveau `stopDepth` et on boucle à nouveau sur la logique d'attente de palier. - La boucle continue ainsi de palier en palier (ex : 24 m → 21 m → 18 m → … → 6 m → 3 m suivant les besoins) jusqu'à ce que `stopDepth` devienne inférieur au dernier palier requis [37].

9. **Remontée finale à la surface :** Lorsque tous les paliers requis sont terminés, il reste à remonter du dernier palier (3 m ou 6 m) vers la surface. Le code prévoit une dernière simulation de cette ascension si nécessaire [38]. En pratique, si le dernier palier est à 3 m, la boucle de paliers s'arrête une fois les 3 m tenus ; la remontée 3→0 m sera simulée ici. On procède par incréments d'1 min à 9 m/min comme avant, en mettant à jour les tissus et en incrémentant le TTS [38].

10. **Sortie des résultats :** La fonction retourne un objet `DecompressionPlan` avec :

11. la profondeur du **premier palier** (`firstStopDepth`),
12. la liste des **paliers** (profondeur en mètres, durée en minutes, et GF effectif) dans l'ordre décroissant de profondeur,
13. le **TTS** total en minutes [39]. Par contrat, le TTS est arrondi à l'entier le plus proche [40]. Les paliers, eux, sont déjà entiers (puisqu'on accumule des minutes entières).

Ce plan est ensuite utilisé par l'UI (cf. `demo/ui/index.html`) pour afficher le **TTS** et la table des paliers [41] [42]. Par exemple, pour une plongée à 40 m pendant 10 min (air, GF 85/85), le programme calculera un petit palier à 3 m, et le TTS affiché inclura le fond, la remontée et ce palier (quelques minutes).

Vérification point par point du modèle Bühlmann ZH-L16C + GF

Compartiments & demi-vies : L'implémentation utilise bien **16 compartiments** pour l'azote et l'hélium, avec les demi-vies standard de ZH-L16C (5 min à 635 min pour N_2, 1.88 min à 240 min pour He) [1]. Ces valeurs correspondent aux références (ZH-L16A/B/C) publiées par Bühlmann. Le modèle considère à la fois N_2 et He dans chaque compartiment, ce qui permet de gérer les gaz trimix correctement.

Coefficients critiques a et b : Les tableaux de coefficients **A_N2, B_N2, A_He, B_He** implémentent les **valeurs ZH-L16C** pour chaque compartiment [43]. Par exemple, pour le compartiment 1 (N_2 5 min) on a A_N2=1.1696, B_N2=0.5578, et pour le compartiment 16 (N_2 635 min) A_N2=0.2327, B_N2=0.9653 [43]. Ces chiffres sont conformes aux tables ZH-L16C diffusées (variant de ZH-L16B plus conservatrice sur les tissus lents [44]). De même, les coefficients pour He (basés sur les travaux de Bühlmann adaptés pour l'hélium) sont présents. *Ainsi, la détermination des pressions maximales tolérées (M-values) est en ligne avec le modèle publié.*

Calcul des pressions inertielles : La fonction de calcul des pressions inertielles inspirées `computePinsp` utilise correctement la formule $P_{insp} = (P_{amb} - P_{H2O}) \times f_{inert}$ [45]. Avec $P_{H2O}=0,0627$ bar, ceci correspond aux valeurs utilisées par les ordinateurs de plongée (OSTC, etc.) et la littérature moderne, légèrement supérieures à la constante 0,0567 autrefois utilisée à la surface [46]. Le résultat a été validé par un test unitaire dans le dépôt : à 1 bar avec 79% N_2, on obtient ~0,740 bar de PN_2 alvéolaire [47], ce qui correspond bien à $(1,013-0,0627)\times0,79 \approx 0,75$ bar attendu (les ~0,01 bar d'écart étant dus aux arrondis acceptés dans le test).

Formule Gradient Factor (GF) : L'algorithme applique la formule d'**Erik Baker** pour moduler la pression ambiante minimale tolérée via un gradient factor [10]. La formule implémentée ${P_{amb,min} = (P_{tiss} - GF \cdot a) / (GF/b + 1 - GF)}$ est identique à celle figurant dans l'article de Baker [13] et la documentation d'implémentations reconnues [11]. Cette approche étend les équations de Bühlmann en multipliant le surplus de pression permissible par un facteur GF entre 0 et 1. Un GF de 1 (100%) donne le plafond "brut" de Bühlmann (M-value), tandis qu'un GF plus faible impose un plafond plus profond (plus de sécurité) [14]. **La présence de deux GF (Low/High)** est gérée en interpolant linéairement la valeur du GF en fonction de la profondeur de chaque palier [31]. GF Low s'applique au *premier palier*

(profondeur maximale de déco), GF High à la *surface*. Entre les deux, le GF augmente graduellement à mesure que la profondeur diminue, ce qui se traduit par des paliers profonds plus conservateurs et un relâchement progressif en remontant [31] [48]. *Cette méthode (GF évolutif du plus bas au plus haut) est conforme aux principes du gradient factor tel qu'utilisé par les ordinateurs de plongée modernes : « GF Low détermine la profondeur du premier palier, GF High la valeur en surface, et le GF effectif change continûment pendant la déco »* [15].

Calcul des paliers multiples : Le code gère **plusieurs paliers successifs** de 3 m d'incrément, et pas seulement le dernier palier. La boucle `while` dans `planDecompression` balaye tous les paliers depuis le premier calculé jusqu'au dernier requis [27] [28]. À chaque niveau, le temps d'attente est calculé jusqu'à ce que la condition de montée soit remplie (plafond permissif à 3 m au-dessus) en utilisant le GF interpolé pour le palier suivant [30]. Ainsi, le plan produit peut comporter des paliers profonds, intermédiaires et un palier final, chacun avec sa durée. Par exemple, si une plongée nécessite des paliers à 15 m, 12 m, 9 m, 6 m et 3 m, le programme les inclura tous avec les minutes associées. *Ceci répond à la préoccupation de "prise en compte des multiples paliers et non seulement du dernier" : l'algorithme couvre bien tous les paliers obligatoires à intervalles de 3 m.* De plus, l'option `lastStopDepth` permet de forcer le dernier palier à 6 m au lieu de 3 m si souhaité (certains ordinateurs comme le Shearwater Peregrine imposent 6 m en dernier palier) – le test unitaire le confirme en vérifiant que le dernier stop est bien à 6 m quand l'option est activée [49].

Calcul du NDL/TTS : Le code peut également calculer le **No-Decompression Limit (NDL)** via une fonction (non directement exposée mais logicielle) similaire en fixant GF=100% et en trouvant le temps max sans pallier [50]. Pour le **TTS (Time To Surface)**, la valeur retournée `tts` est censée représenter le temps qu'il faut pour remonter à la surface une fois la plongée entamée. Dans l'implémentation actuelle, `tts` est initialisé à 0 puis **incrémente à chaque minute** simulée pendant la descente, le fond, les paliers et les remontées [19] [22] [34]. À la fin, `tts` est arrondi et renvoyé [40]. *Attention :* tel que codé, le TTS inclut en réalité **toute la durée de plongée** (descente + fond + déco). Habituellement, dans le jargon plongée, "TTS" désigne plutôt le temps pour regagner la surface **à partir de maintenant** (typiquement, à partir du début de la remontée). Nous reviendrons sur ce point d'interprétation plus loin. En l'état, si l'on veut le TTS post-fond, il faudrait soustraire le bottom time. Hormis ce détail de définition, le calcul du TTS se met bien à jour à chaque étape cohérente du profil.

En somme, l'implémentation couvre fidèlement le modèle Bühlmann ZH-L16C avec Gradient Factors. Les calculs de saturation/désaturation sont corrects, les coefficients appropriés, la formule de gradient factor est conforme aux publications [11], et l'algorithme gère l'enchaînement des paliers de manière itérative comme attendu. Les tests unitaires fournis confirment d'ailleurs des comportements clés : par exemple, une plongée **40 m/10 min Air GF85/85** nécessite bien désormais un palier (alors qu'une formule erronée antérieure ne le faisait pas) [51], et l'option de dernier palier 6 m fonctionne [49].

Problèmes et imprécisions relevés

Malgré la qualité de l'implémentation, quelques **anomalies ou points perfectibles** ont été identifiés :

Aspect / Fonctionnalité	Observation / Anomalie	Conséquence sur la précision du plan
Définition du TTS (Time To Surface)	Le code cumule le TTS depuis le départ de la plongée, y compris descente et temps de fond [19] [22]. Or, en pratique, le TTS désigne plutôt le temps *nécessaire pour remonter* une fois le profil en cours (fond terminé). Actuellement, si le plongeur a 20 min de fond et 10 min de déco, le programme affichera TTS≈30 min.	**Confusion potentielle** – Le TTS affiché par l'outil n'est pas le même indicateur que celui d'un ordinateur de plongée (qui n'inclurait pas le temps passé au fond). Pour l'utilisateur, cela peut prêter à confusion dans la lecture du plan.
Arrondi des durées (granularité 1 min)	L'algorithme simule les segments de plongée par pas d'une minute entière. Il arrondit la durée de certaines phases à l'entier supérieur via `Math.ceil` (ex. remontées) [26], et accumule des minutes entières pour les paliers. Il n'y a pas de gestion de fractions de minute.	**Légère surévaluation des durées** – Cette discrétisation introduit un conservatisme marginal : on peut surestimer un palier ou une remontée de quelques dizaines de secondes au plus. Sur plusieurs paliers, le décalage peut s'additionner (quelques minutes d'excès total possible). Par rapport à un plan calculé en continu (ex. Subsurface ou tables), le profil généré pourrait être un peu plus long.
Option dernier palier 6 m	Lorsqu'on force le dernier palier à 6 m (`lastStopDepth:6`), le code respecte bien cette contrainte [52]. Cependant, il faut noter que le modèle Bühlmann n'exige pas intrinsèquement de palier à 6 m plutôt qu'à 3 m ; c'est un choix de configuration (par ex. pour utiliser de l'oxygène pur à 6 m).	**Choix de configuration** – L'implémentation est correcte, mais l'utilisateur doit être conscient que sélectionner 6 m comme dernier palier modifie légèrement la façon dont GF_high s'applique (on ne montera pas à 3 m). Ce n'est pas un problème de précision, plutôt de paramétrage en fonction des pratiques (ex. plongeurs tech vs loisir).
Gradient Factor unique par palier	Le calcul de l'autorisation de quitter un palier se base sur le GF interpolé du **palier suivant** (`gfNext`) [30]. Cela fonctionne bien, mais signifie qu'au palier courant on n'utilise pas explicitement le GF correspondant à cette profondeur-là pour évaluer la sursaturation restante.	**Négligeable** – En pratique, cette logique "par palier suivant" est équivalente à d'autres approches (certains algorithmes recalculent un plafond « courant » à GF intermédiaire et voient s'il passe sous 3 m, revient au même). Ce choix n'affecte pas la précision si l'interpolation GF est linéaire. C'est un détail d'implémentation plus qu'une erreur.

Aspect / Fonctionnalité	Observation / Anomalie	Conséquence sur la précision du plan
Absence de suivi de la toxicité O_2	Le planificateur ne calcule pas les expositions à l'oxygène (CNS % ou OTU). Actuellement, pour de l'air (21% O_2) ce n'est pas critique, mais si l'utilisateur planifie au Nitrox ou oxygène pur en palier, il n'y a pas d'alarme sur les limites d'oxygène.	**Limitation fonctionnelle** – Contrairement à des outils complets (Subsurface, MultiDeco, etc.), ici une plongée à forte FO_2 pourrait dépasser les limites CNS sans indication. Pour un plan **"précis"**, il faudrait intégrer ces calculs afin d'éviter de suggérer un plan dangereux du point de vue oxygène.
Pas de gestion multi-gaz	L'algorithme suppose un seul gaz du début à la fin. Aucune gestion de changement de mélange (par ex. passage à un Nitrox ou oxygène en déco).	**Plans moins optimisés** – En l'état, le planificateur calculera la déco comme si on restait au gaz fond tout du long. Dans la réalité, les plongeurs utilisent des gaz enrichis pour raccourcir la déco. Sans cette fonctionnalité, le plan peut être **plus long qu'il ne pourrait l'être** avec changements de gaz, et ne reflète pas les pratiques avancées.
Modèle statique (pas de facteurs externes)	Le modèle Bühlmann+GF ne tient pas compte de facteurs comme l'effort, la température, la prédisposition individuelle, etc., qui pourraient justifier d'ajuster le profil.	**Non personnalisable** – Ce n'est pas un "défaut" du code en soi (c'est inhérent au modèle Bühlmann). Mais d'autres modèles ou ordis ajoutent des **facteurs de conservatisme** supplémentaires ou des réglages (par ex. +% sur les temps de palier, modèle RGBM conservatif, etc.). Ici seul le choix des GF permet d'ajuster la sécurité.

En résumé, les écarts relevés n'indiquent pas de fautes de calcul majeures, mais plutôt des choix d'implémentation à clarifier ou des fonctionnalités absentes qui limitent la **précision globale du plan** vis-à-vis d'un logiciel professionnel. La plus notable est le calcul du TTS qui pourrait induire l'utilisateur en erreur sur son interprétation. Les autres points (arrondis, absence de multi-gaz, etc.) jouent sur la précision ou la conformité aux meilleures pratiques, sans invalider le cœur de l'algorithme.

Recommandations pour améliorer la précision du planificateur

Pour rendre ce planificateur aussi précis et fiable que des références comme Subsurface, voici **plusieurs pistes d'amélioration** concrètes :

1. Corrections de l'implémentation existante

- **Redéfinir le calcul du TTS :** Il est recommandé d'afficher le TTS comme le *temps de décompression à partir du moment où l'on quitte le fond*. Actuellement, on peut soit :
- **Soustraire** le temps de fond (et de descente) du `tts` avant affichage, soit
- **Mieux** : scinder le calcul en deux variables – par ex. `totalDiveTime` (tout cumulé) et `decoTime` (seulement la déco + remontée). De cette façon, on peut présenter à l'utilisateur les deux informations (durée totale de plongée prévue **et** TTS à partir du fond). Cela éviterait

l'ambiguïté. Subsurface par exemple affiche généralement la durée totale du profil et la durée de déco séparément.

- **Augmenter la résolution temporelle :** Pour gagner en précision, envisager de **réduire le pas de calcul** en dessous de 1 min, au moins pour certaines phases. Par exemple, on pourrait utiliser un pas de 30 s (ou même 10 s) lors des remontées et paliers, afin de ne pas systématiquement arrondir à la minute supérieure. Certes, l'affichage final restera à la minute, mais cela permettrait de déterminer qu'un palier de 2,4 min peut être arrondi à 2 min au lieu de 3. Une autre approche plus complexe serait d'employer directement les formules continues : par ex. calculer le temps exact qu'il faut à un compartiment pour passer sous le seuil au palier suivant (en résolvant l'équation exponentielle pour Δt). Cependant, une **intégration plus fine par pas plus courts** est plus simple à mettre en œuvre et suffirait pour coller aux résultats d'autres logiciels (qui souvent calculent à la seconde près). En bref, passer à une simulation en seconds ou décimales de minute améliorerait la fidélité du profil.

- **Vérifier l'alignement des constantes avec les références :** Les coefficients utilisés semblent corrects. Il serait bon de les comparer aux sources officielles ou à la documentation de Subsurface. Par exemple, Subsurface utilise par défaut ZH-L16C avec PH_2O=0,0627 bar, ce qui est cohérent avec notre implémentation [53] [54]. S'assurer que le choix de PH_2O=0,0627 est le même dans Subsurface (normalement oui pour les plongées en eau salée, 37°C). De même, confirmer les valeurs de demi-vie et M-values. Une différence minime comme 0,4701 vs 0,4710 sur un coefficient *a* est insignifiante, mais autant corriger pour être rigoureusement exact. On peut trouver ces tableaux dans les publications ou la documentation du code source de Subsurface ou de la bibliothèque de calcul (e.g. `deco.c` dans Subsurface).

- **Nettoyage du code :** Quelques duplications mineures sont présentes (p. ex. `updateConstantDepth` est défini dans `utils.ts` **et** redéfini dans `algorithm.ts`). Pour éviter toute incohérence, il vaudrait mieux avoir une seule fonction canonique. Idem pour l'initialisation des tissus (présente deux fois). Cela n'impacte pas le résultat mais la maintenance.

2. Tests de validation croisée

Pour garantir que le plan est « précis comparable à Subsurface », il faut le **valider sur des cas concrets**. Actions suggérées :

- **Comparer avec Subsurface ou des tables publiées** : Prendre plusieurs profils types (plongée à l'air 30 m 25 min, 40 m 10 min, déco Nitrox, etc.) et générer le plan avec notre outil puis avec Subsurface (en configurant les mêmes GF). Vérifier que les paliers et durées correspondent à 1 min près. Toute divergence notable serait le signe d'un détail manquant. Par exemple, Subsurface intègre-t-il une "arrivée sur le palier" instantanée ou en continu ? Si des écarts sont constatés, ajuster le calcul (p. ex. l'overshoot des arrondis mentionné plus haut).

- **Utiliser une bibliothèque de référence** : Il existe des bibliothèques open-source de calcul déco, telles que **DecoTengu (Python)** ou **libbuhlmann (C)**, qui implémentent Buhlmann ZH-L16+GF. On pourrait écrire des tests automatisés utilisant ces libs en parallèle pour vérifier que `planDive` donne les mêmes résultats. Par exemple, DecoTengu permet de calculer un profil complet en quelques appels ; on pourrait l'utiliser sur nos profils de test et comparer résultats (paliers, TTS). C'est un excellent moyen de détecter de subtiles différences de formule (DecoTengu est bien documenté et réputé correct [11] [32]).

- **Tests extrêmes** : Tester des cas aux limites (plongée NDL juste avant déco, plongée très longue avec de multiples paliers profonds, etc.). Ainsi, on s'assure que l'algorithme tient en termes de performance et de cohérence. Par exemple, une plongée 50 m 60 min sur Trimix avec GF 20/85 produira de nombreux paliers profonds ; vérifier que l'algorithme les calcule sans "oublier" un palier (un piège classique serait un compartiment lent qui impose un léger stop à mi-profondeur alors que plus rien ne l'indique autrement – normalement couvert par la formule GF Low).

- **Régression tests** : Conserver les tests existants (qui valident la correction de la formule Baker dans cette implémentation – ceux-ci indiquent que l'algorithme est maintenant plus conservateur, ce qui est positif [51]). En ajouter de nouveaux pour les points corrigés (ex. un test que TTS n'inclut plus le fond si on change cette logique).

3. Gestion de l'oxygène et des gaz multiples

Pour un planificateur vraiment **complet**, il est nécessaire d'élargir le scope au-delà du strict modèle de dissolution inertielle :

- **Calcul de la toxicité oxygène (CNS/OTU)** : Intégrer un module calculant le %CNS (Central Nervous System Oxygen Toxicity) et les OTU (Oxygen Tolerance Units) en fonction du temps passé à chaque pression partielle d'O_2. Subsurface et les autres logiciels utilisent les standards NOAA ou Bühlmann pour cela. Par exemple, exposer 15 min à pO_2 1,6 bar correspond à 15÷(dat*unit) ~ X% CNS, etc. On peut coder une fonction qui, étant donné FO_2 du gaz et profondeur, calcule pO_2 et incrémente un compteur CNS par minute. Idem pour OTU (le modèle NOAA 1991 donne une formule d'OTU par minute au-delà de 0,5 bar pO_2). Ensuite, afficher ces résultats ou alerter si on dépasse 100% CNS ou les limites journalières d'OTU. Bénéfice :** cela rendra le plan comparable à ce qu'indiquent les ordis de plongée (p. ex. un plan oxy à 6 m pendant 15 min fera apparaître une alarme CNS ~ 80%). C'est crucial en plongée tek pour éviter les accidents hyperoxiques.

- **Support des changements de gaz (multi-gaz)** : Implémenter la possibilité de spécifier plusieurs gaz et des segments de plongée. Par exemple, le plongeur peut vouloir : "40 min à 50 m sur Tx 18/45, puis remontée, switch à Nitrox 50 à 21 m, puis O_2 à 6 m". Aujourd'hui, on ne peut entrer qu'un gaz. Il faudrait donc permettre à `planDive` de prendre une liste de segments (profondeur, durée, gaz) plutôt qu'une seule profondeur/durée. La décompression se calculerait alors en enchaînant ces segments tout en continuant à suivre les tensions de chaque compartiment. Concrètement : on appliquerait `updateConstantDepth` sur chaque segment successivement, et après le dernier segment "fond", on exécuterait la procédure de déco. Lors des paliers, on devrait aussi permettre le **changement de gaz au palier** s'il y a un gaz de déco disponible (ex. switch sur O_2 à 6 m pour accélérer la désaturation). Il s'agit certes d'une extension non triviale, mais c'est ce qui distingue un plan **basique** d'un plan **expert**. Des références comme **MultiDeco** ou **V-Planner** proposent ces fonctionnalités. Il faudra ajouter une logique pour déterminer automatiquement le meilleur gaz à utiliser à chaque palier (ou laisser l'utilisateur le spécifier). Par exemple, Subsurface permet de saisir des "changements de gaz planifiés" et calcule en conséquence. En implémentant cela, notre agent deviendrait bien plus robuste et réaliste pour des plongées techniques.

- **Vérifier l'effet sur les tissus lors des switches** : un point technique sera de recalculer le plafond à chaque fois qu'on change de gaz, car la composition du gaz respiré n'affecte pas directement le calcul du plafond (celui-ci dépend des tensions tissulaires actuelles), **mais** l'évolution future des tensions (donc la durée de palier restante) va changer. En pratique, l'algorithme existant, une fois modifié pour multi-gaz, pourra simplement continuer la boucle de paliers en utilisant le gaz actif du palier (ex. si O_2 pur dispo à 6 m, alors FN_2=0 → plus de charge N_2 ajoutée pendant ce palier,

d'où désaturation accélérée). Il faudra donc choisir le gaz "optimal" pour chaque palier (typiquement le plus riche en O_2 toléré à cette profondeur).

4. Incorporer les meilleures pratiques et autres modèles

- **Gradient Factors recommandés :** S'assurer que l'utilisateur comprend bien le choix des GF. Par défaut, on pourrait fournir des couples GF adaptés au type de plongée : *par ex.* GF 30/75 ou 40/85 pour des plongées profondes (réduction du risque microbulles dans les tissus lents [55]), vs GF 85/85 pour plongée no-stop loisir (maximiser le temps fond tout en restant dans les M-values). Des travaux récents (Dr Simon Mitchell, 2018) suggèrent d'éviter les GF_low trop bas (du style 20/85) car des "deep stops" excessifs pourraient sous-désaturer les compartiments lents et augmenter le risque de syndrome tardif. Il serait judicieux de documenter ou guider le réglage des GF. *Exemple :* on peut inclure dans la documentation ou l'UI un encadré : "GF bas 30/70 conseillés pour trimix > 40 m d'après Mitchell, GF80/80 pour plongée à l'air modérée" [56] [55].

- **Validation par d'autres modèles :** Le modèle Bühlmann GF est éprouvé, mais on peut envisager de le confronter à des modèles alternatifs pour améliorer la sécurité. **Subsurface** propose aussi le modèle **VPM-B** (Varying Permeability Model**)**, un modèle à bulles qui préconise des paliers plus profonds (deep stops) basés sur la croissance des microbulles. Certains plongeurs utilisent VPM ou RGBM (Suunto) pour plus de conservatisme en profondeur. Il pourrait être intéressant d'**intégrer un second algorithme** (même simplifié) pour comparer les résultats. Par exemple, calculer un plan VPM-B en parallèle et signaler si celui-ci exigerait des paliers plus profonds/longs que Bühlmann GF – cela servirait de *cross-check*. Cependant, implémenter VPM-B n'est pas anodin (c'est un modèle différent, avec des paramètres propres comme le "crush pressure", etc., et souvent il est sous licence). À défaut, on peut emprunter des idées : par ex. ajouter en option des *paliers profonds de sécurité* (un peu comme certains plongeurs font un stop empirique à mi-profondeur même si non requis). Mais attention, l'intérêt de ces deep stops additionnels a été remis en question par des études récentes.

- **Modèle probabiliste (BVM3)** : La mention du modèle BVM(3) dans la question ouvre la porte à une approche très avancée. BVM3 est un **Bubble Volume Model** de 3 compartiments avec fonction de risque probabiliste, développé notamment dans le cadre des recherches de l'US Navy [57] [58]. En gros, il cherche à prédire la probabilité de DCS en modélisant la croissance des bulles microscopiques et en la calibrant sur des données empiriques. C'est un modèle qui permet non plus de dire "palier requis / non requis" de façon déterministe, mais "X% de risque si on suit tel profil". Intégrer BVM3 dans un planificateur serait complexe : il faudrait des données et le code du modèle (probablement disponible dans certaines publications ou brevets [59]). **Recommandation :** Ce n'est sans doute pas indispensable pour un planificateur opérationnel, car Bühlmann+GF, bien utilisé, donne déjà de bons résultats pratiques. Cependant, si le but est d'avoir un outil de recherche ou de vérification, on pourrait implémenter un **calcul de risque estimé**. Par exemple, des travaux (Kevin Watt et al., 2019) ont utilisé les modèles probabilistes pour comparer des profils deep-stop vs shallow-stop. On pourrait ajouter en sortie un **indice de risque** basé sur BVM3 ou autre, pour chaque plan. Mais c'est un projet en soi. Une alternative plus simple est de fournir un facteur de sécurité supplémentaire : ex. afficher "Surfacing GF" : le GF actual (appelé parfois "GF99" sur les ordinateurs Shearwater) du compartiment leader en fin de déco. S'il est très proche de 100%, le profil est tendu ; s'il finit à 70%, il est conservateur. C'est déjà un indicateur de "marge de sécurité".

- **IA pour validation dynamique** : L'idée d'utiliser des modèles d'**Intelligence Artificielle** pour valider ou ajuster le plan est novatrice, mais encore exploratoire. On pourrait imaginer entraîner un réseau de neurones sur des milliers de profils plongée connus (avec ou sans accident) afin de

prédire le risque ou de conseiller des ajustements de profil. Toutefois, les données réelles de plongée avec DCS sont limitées, et les modèles physiques restent la référence. Un champ possible d'application de l'IA serait de **personnaliser** le modèle à l'individu : par ex. ajuster les demi-vies ou GF en fonction de l'historique de plongée d'une personne (certains cherchent à le faire, mais rien de concret n'est encore disponible publiquement). Pour l'instant, une approche prudente serait d'utiliser l'IA pour vérifier la cohérence du plan proposé en temps réel : par exemple un agent d'IA pourrait surveiller les outputs (paliers, TTS, CNS) et les comparer à des règles apprises (ou des plans calculés par d'autres algos) et signaler s'il voit un écart anormal. Mais cela revient à coder ces règles explicitement dans notre cas. **En résumé**, l'IA n'est pas encore un outil éprouvé pour "corriger dynamiquement" un plan de plongée – la meilleure validation reste la comparaison avec des standards établis et l'avis d'experts humains. On peut citer un fil de discussion récent où des plongeurs ont interrogé ChatGPT sur les GF à utiliser : les réponses bien que littérairement correctes se sont avérées datées ou imprécises par rapport aux recommandations actuelles [60] [55], ce qui incite à la prudence.

5. Documentation et références externes

Pour améliorer le planificateur et rassurer sur sa fiabilité, on peut s'appuyer sur des **références reconnues** :

- **Publications de base** : Fournir dans la documentation du projet des références comme l'article d'Erik Baker *"Clearing Up The Confusion About Deep Stops"* [61] [62] et *"Understanding M-Values"*, qui expliquent la genèse des GF et les principes du modèle. Ces lectures permettront à l'utilisateur avancé de comprendre les tenants du calcul. Idem, référencer le manuel du **DRV OM** (US Navy Diving Manual) ou les tables Bühlmann originales peut être utile.

- **Code source exemplaire** : Mentionner ou s'inspirer du code de **Subsurface** (open-source, C++) pour la planification. Subsurface implémente ZH-L16C+GF, VPM-B, et gère multi-gaz, altitude, etc. Parcourir son code pourrait donner des indications sur des détails (par ex. comment ils gèrent l'arrondi des paliers, la précision temporelle, ou la transition GF Low→High exactement). De même, le projet **DecoTengu** (Python) est bien documenté et pourrait servir de référence algorithmique (leur documentation mathématique a été citée plus haut pour valider nos formules [11] [32]).

- **Outils de planification existants** : MultiDeco (logiciel commercial) et PastoDeco, par exemple, offrent des comparaisons entre modèles. On peut recommander à l'utilisateur de croiser les résultats de notre planificateur avec ceux de ces outils dans un but de vérification. Si des divergences apparaissent, cela peut signaler soit un bug, soit un paramètre différent (par ex. MultiDeco par défaut utilise VPM-B, donc il faut comparer en mode Bühlmann pour être cohérent).

- **Guides et manuels** : Inclure des liens vers des articles pédagogiques comme celui de Dive Rite sur les Gradient Factors [63] [64], ou des tutoriels (plongeur techno) expliquant comment choisir ses GF et gaz. Cela aidera l'utilisateur final à utiliser correctement l'outil et à interpréter les résultats. Par exemple, l'article DiveRite cité explique très bien le concept de plafond, de sursaturation et l'effet des GF [65] [48].

En appliquant ces recommandations, le **planificateur Bühlmann+GF** gagnera en exactitude et en fonctionnalités, devenant un véritable outil de planification comparable aux références du domaine. Il combinera la rigueur scientifique (validation par publications et code reconnu) et la prise en compte des **bonnes pratiques plongée** (conservatisme modulable, gaz adaptés, surveillance O_2, etc.).

Enfin, il convient de tester toute modification avec une attention particulière à la **sécurité** : un planificateur se doit d'errer du côté conservateur en cas de doute. Chaque amélioration (par ex. réduire les arrondis ou ajouter un deep stop optionnel) devra être évaluée quant à son impact sur le risque. L'objectif ultime – partager par l'utilisateur – est d'obtenir des plans *précis, fiables et personnalisables* pour plonger en toute sécurité, en s'appuyant sur le meilleur des connaissances actuelles en décompression [14] [48].

Sources : L'analyse ci-dessus s'est appuyée sur le code du dépôt fourni, les articles d'Erik Baker sur les Gradient Factors [13], la documentation de la librairie DecoTengu [11] [32], des discussions de la communauté plongée technique [60] [55], ainsi que des ressources pédagogiques sur les modèles de décompression (DiveRite, DAN, etc.) [64] [48]. Ces références confirment la validité des formules utilisées et ont inspiré les pistes d'amélioration proposées.

---

1 2 43 constants.ts
https://github.com/EpicReality3/buhlmann-planner-agent/blob/7d5f0de4c8f5659edb84f92fb7ca090a693e2a62/src/core/constants.ts

3 5 6 7 45 utils.ts
https://github.com/EpicReality3/buhlmann-planner-agent/blob/7d5f0de4c8f5659edb84f92fb7ca090a693e2a62/src/core/utils.ts

4 index.ts
https://github.com/EpicReality3/buhlmann-planner-agent/blob/7d5f0de4c8f5659edb84f92fb7ca090a693e2a62/src/adapter/index.ts

8 9 11 17 32 44 53 54 Decompression Model — decotengu 0.14.1 documentation
https://wrobell.dcmod.org/decotengu/model.html

10 12 16 18 19 20 21 22 23 24 25 26 27 28 29 30 31 33 34 35 36 37 38 39 40 algorithm.ts
https://github.com/EpicReality3/buhlmann-planner-agent/blob/7d5f0de4c8f5659edb84f92fb7ca090a693e2a62/src/core/algorithm.ts

13 61 62 citeseerx.ist.psu.edu
https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=eb03802361c6050ec2d0bb250c3db2407602032e

14 15 48 63 64 65 Gradient Factors - Dive Rite | Equipment for Serious Divers
https://diverite.com/gradient-factors/

41 42 index.html
https://github.com/EpicReality3/buhlmann-planner-agent/blob/7d5f0de4c8f5659edb84f92fb7ca090a693e2a62/demo/ui/index.html

46 pydplan_buhlmann.py
https://github.com/eianlei/pydplan/blob/08501a1e67e091fc0080aca0f4f9462efd07e8bf/pydplan_buhlmann.py

47 49 51 52 test.ts
https://github.com/EpicReality3/buhlmann-planner-agent/blob/7d5f0de4c8f5659edb84f92fb7ca090a693e2a62/tests/test.ts

50 ZHL16.java
https://github.com/Jens-Horstmann/Buhlmann-ZHL-16/blob/70540300b2026f1df38b21ab4dac4859d13ffe1f/src/main/java/BuhlmannZHL16/ZHL16.java

55 56 60 Using AI (Artificial intelligence) for dive planning | ScubaBoard
https://scubaboard.com/community/threads/using-ai-artificial-intelligence-for-dive-planning.654577/

57 [PDF] Dual Phase Decompression Theory and Bubble Dynamics
https://www.divetable.info/skripte/Bubble_Dynamics_02.pdf

58 59 Method and device for predicting risk of decompression sickness
https://patents.google.com/patent/US7474981B2/en